The Resource Journal for IBM Micro Programmers

# PROGRAMMER'S Journal

SEPTEMBER/OCTOBER 1988
Volume 6.5
$5.00 Canada $6.25

C++
on OS/2

Ada
& Software
Engineering

REXX
on the PC

VGA
Text Paging

Dynamic Arrays
& Records in TP4

A Presentation Manager
Template in C

WATCOM C
& the ANSI C Standard

# 21609***** 4**6MARAPR89
RON K UNZ
WALL STREET ANALYTICAL INC
33-41 85TH ST
JACKSON HEIGHTS NY 11372

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Make your programs millions of times smarter.

More and more, programmers and workstation builders are using DESQview 2.0 as a development tool. The reason is simple. They can create powerful, multitasking solutions today for the millions of DOS PCs in use today. Solutions comparable to those promised for tomorrow by OS/2.

Some of the applications under development right now using DESQview 2.0 API Tools: CAD, medical systems, insurance, 3270 mainframe communications, network management, real estate, typesetting, point of sale, education, commodity trading, stock trading and online voting.

## The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get access to the powerful capabilities built into DESQview--multitasking, windowing, intertask comunications, mailboxes, shared programs, memory management, mousing, data transfer, menu-building and context sensitive help.

## Bells and Whistles

A program taking advantage of the DESQview 2.0 API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask communications to rapidly exchange data between programs, share common code and data, or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

## 80386 Power

80386 programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview--side by side with other 80386 and DOS programs. The breakthroughs tha make this possible: DOS Extenders from PharLap Software and AI Architects and DESQview support of these DOS extenders

## DESQview Developer Support

So if you are a developer, looking to create programs with mainframe capabilities, but wanting to sell into the existing base of millions of DOS PCs, we can be very helpful. We offer DESQview API Tools for assembler or C programmers: a debugger, a Reference Manual. And lots more tools on the way.

Plus, we offer DESQview API Developers Conferences. Come learn about the DESQview 2.0 API and 80386 DOS Extenders. Meet 80386 experts as well as those smart people who are creating DESQview 2.0 API workstation solutions.

For more information, call or write us.

# Bringing New Power to DOS. DESQview 2.0 API Toolkit.

Quarterdeck

Quarterdeck Office Systems 150 Pico Blvd.,Santa Monica, CA 90405
(213) 392 9851

Number 147 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Simply the BEST C and Pascal on AT, 386, Sun, Apollo, RT, VAX, 370

"The most rock-solid C compiler in the industry. Superb technical support and portability. Superior code generated."
**Gordon Eubanks, Symantec — Q&A (386).**

"It simply works, with no trouble, no chasing strange bugs, and excellent warning and error messages ... a professional product."
**Robert Lerche, Bay Partners.**

"For large-scale software development, the highest quality C compiler available on the market today. Pragmas are great. Quality of support is exceptional." **Randy Neilsen, Ansa—Paradox (DOS,OS/2).**

"15% smaller and 15% faster than Lattice C."
**Robert Wenig, Autodesk.**

"Our software is running anywhere from 30 to 50% faster than when compiled under Lattice." **David Marcus, Micronetics.**

"We switched from Lattice due to a 10% reduction in code size. The compiler is very stable." **Lee Lorenzen, Ventura Software — Ventura Publisher, marketted by Xerox Corp.**

"Best quality emitted code by any compiler I've encountered. Often amazing." **Bill Ferguson, Fox Software — FoxBase (386).**

"Messages sometimes pointed out type mismatches, incorrect-length argument lists, and uninitialized variables that had been undetected for years [in 4.x bsd]." **Larry Breed, IBM ACIS [RT PC].**

"Diagnostics turned up bugs missed by other compilers. Rapid bug fixes by technical support, someone who knew what he was talking about. 80386 code is well optimized."
**Tim Addison, Logistics Data Systems.**

"386 protected mode support is fantastic, especially the access to large amounts of memory. It's mainframe compute power on a PC." **Dan Eggleston, Viewlogic.**

"The preprocessor supplied with Professional Pascal is quite useful. The code quality and control over segmentation and memory models are superior to MS Pascal." **Bob Wallace, QuickSoft.**

## Check Out These Reviews

- **High C ™:**

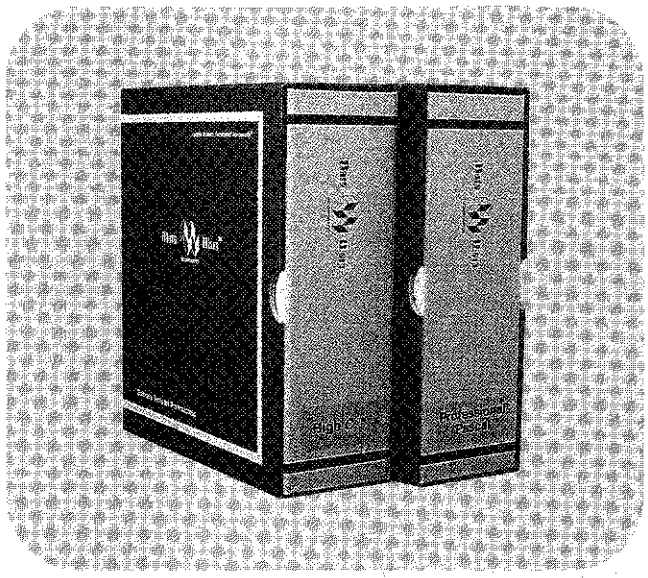
| | | |
|---|---|---|
| *Computer Language* | February 1986, '87 | |
| *Dr. Dobb's Journal* | August 1986 | |
| *PC Magazine* | Jan. 27, 1987 | (80386 version) |
| *Dr. Dobb's Journal* | July 1987 | (80386 version) |
| *BYTE Magazine* | November 1987 | (80386 version) |

- **Professional Pascal ™:**

| | | |
|---|---|---|
| *PC Magazine* | Dec. 29, 1985 | |
| *Computer Language* | May 1986 | |
| *PC Tech Journal* | July 1986 | |
| *Journal of Pascal, Ada, & Modula-2* | Nov.-Dec. 1986 | |
| *BYTE Magazine* | Dec. '86, June '87 | (80386 version) |

## Why MetaWare compilers

- They are specifically designed for serious software developers.
- They are reliable and robust: they don't break at every turn.
- Their generated code is the best, or near best, on each architecture.
- Their superior diagnostic messages help you produce better products more quickly.
- Your source can be ported with ease to the most popular systems.
- You can link mixed-language modules from our compilers, others
- You can benefit from high-level, personal technical support.
- You can take advantage of the latest ANSI C extensions, and/or extensive Pascal extensions. **High C** has been tracking the ANSI Standard for two years; **Professional Pascal** will soon have a VS Pascal compatibility switch and several Apollo Pascal ext'ns.
- You can take advantage of the latest 387 and Weitek 1167 support — we have the only compilers with Weitek real mode support.



## Power Tools for Power Users

Ashton-Tate: dBase III Plus, MultiMate; Autodesk: AUTOCAD, AUTOSKETCH (8087, '387, Weitek); Boeing Computer Services (Sun); CASE Technology (Sun); CAD/CAM giant Daisy ('86, '386, VAX); Deloitte Haskins & Sells; Digital Research: FlexOS; GE; IBM: 4.3/RT, 4680 OS; Lifetree Software (Pascal): Volkswriter Deluxe, GEM-Write; Lugaru: Epsilon; NYU: Ada-Ed cmplr; Semantec: Q&A; Sky Computers; ... (Product names are trademarks of the companies indicated.)

## Industrial-Strength

MetaWare C and Pascal compilers are designed for professional software developers. These tools are loaded with options to control them for special purposes. You can adjust the space-time trade-off in code quality. You can adjust external naming conventions to agree with linkers and operating systems. You can specify segment names for segmented architectures, and to help place code or data in particular places for embedded applications. You can select from five memory models for the 8086 family. And on and on.

## A Partial List of Optimizations

Common subexpression and dead-code elimination, retention and reuse of register contents, jump-instruction size minimization, tail merging (cross jumping), constant folding, short-circuit evaluation of Boolean expressions, strength reductions, fast procedure calls, automatic mapping of variables to registers (where advantageous), ...

## "Platform" — Code Quality

| | |
|---|---|
| Sun,Apollo,SGI | — 18%, 3%, 26% > resident compiler (Dhrystone). |
| PC: DOS, OS/2 | — 3-10% > Microsoft C; 30% > MS Pascal, LatticeC. |
| 386 32-bit DOS | — no competitors, since November, 1986. |
| 286, 386 UNIX | — 66% better than pcc (Dhrystone, 386). |
| VAX VMS | — ≈ DEC's excellent C and Pascal; better features. |
| VAX Ultrix | — 19% > pcc (Dhrystone); much > Berkeley Pascal. |
| RT PC/4.3bsd | — 89% > the original port of pcc (Dhrystone). |
| 370 CMS,UNIX | — much better than any C, and VS Pascal. |
| AMD 29000 | — >40,000 Dhrystones! Available in Q2, cross. |

**(408) 429-6382, telex 493-0879.** ***Since 1979.***

MetaWare™ INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

## The Clear Choice for Large Programming Projects — PC Tech J.

© 1987 MetaWare Incorporated. MetaWare, High C, Professional Pascal, and DOS Helper are trademarks of MetaWare Incorporated. Others and their owners are duely respected.

Number 45 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers

# PROGRAMMER'S Journal

Volume 6.5
September/October 1988

## Contents

**4 Notes from the Editors**

**6 Letters**
Gordon Arbeitman, John P. Brannigan, Dennis Reinhardt & Chuck Gilmore, Richard Allen

**12 PJ Forum:** Computers in the Fast Lane *by Hal W. Hardenbergh*
Hal takes a hard look at the extravagant claims of computer vendors

**16 Consulting:** No-win Jobs and Too Much Work *by Paul Barkley*
Classic consulting dilemmas and some options

**22 On Graphics:** VGA Text Paging *by Michael Abrash*
Find 32 text pages—and assembler routines that should take care of most of your text paging needs

**34 Pascal:** Programming on the Variant Express *by Tom Swan*
Using dynamic arrays and records in TP4, with a graphics code example

**44 C:** An OS/2 Presentation Manager Application Template *by William S. Hall*
A follow-up to last issue's Windows template—the minimum steps required to display a window

**57 C:** Watcom C and the ANSI Standard *by Rex Jaeschke*
WATCOM C V6.0 conformance to the ANSI draft

**60 C++:** Object Oriented OS/2 *by Vaughn Vernon*
C++ comes into its own on a large, complex operating system

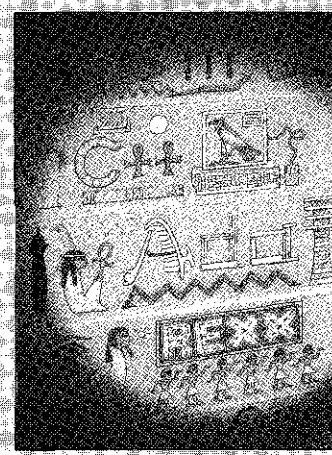
**68 Ada:** Ada—A Software Engineering Tool *by Richard D. Riehle*
No more waivers—the DoD, NASA, the FAA, *et al*, mandate maintainable code

**78 REXX:** The REXX Language on the PC *by Andrew J. Chalk*
IBM's underground utility language surfaces as their standard "scripting language" from PC to mainframe

**87 Miscellaneous** *by M. Steven Baker*
Editors and tools, debuggers, and translators

**95 PJ Billboard**

**96 Advertiser Index**



Object oriented and system utility languages emerge as we move into software engineering and increasing standardization across computer models. *Artist Ralph Bentley*

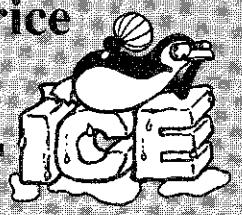


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# SERIOUS DEBUGGING *at a* REASONABLE PRICE

## All the speed and power of a hardware-assisted debugger at a software price

**Soft-ICE**



### Hardware-level break points

REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

### Break out of hung programs

With a keystroke - no external switch necessary. Even with interrupts disabled.

### Breaks the 640K barrier

Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

### Works with your favorite debugger

Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

### *Solve tough systems problems too*

Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

### How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

**"Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have."**

*Dr. Dobb's Journal — May 1988*

## RUN CODEVIEW IN ONLY 8K!

**MagicCV**

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

### Don't let 640K be your limit!

If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

### Don't let the debugger hide the bug!

Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

### How MagicCV works

MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

### MagicCV is easy to use

If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

Save $86

| | |
|---|---|
| MagicCV | $199 |
| Soft-ICE | $386 |

***Buy Both and Save $86!***

**CALL TODAY**
**(603) 888 - 2386**
***or FAX* (603) 888 - 2465**

30 day money-back guarantee
Visa, Master Card and AmEx accepted

NU-MEGA TECHNOLOGIES

P.O. BOX 7607 • NASHUA, NH 03060-7607

### MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

Both require 80386 AT compatible or IBM PS/2 Model 80. MagicCV requires at least 384K of extended memory. CodeView is a trademark of Microsoft Corporation.

Number 160 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The Resource Journal for IBM Micro Programmers
PROGRAMMER'S Journal

**Editor**
Erin O'Connor
**Technical Editor**
M. Steven Baker
**Contributing Editors**
Michael Abrash, Frank D. Greco, Tom Swan

**Contributing Writers**
Michael Abrash
M. Steven Baker
Paul A. Barkley
Andrew J. Chalk
William S. Hall
Hal W. Hardenbergh
Rex Jaeschke
W.G. Madison
Richard D. Riehle
Tom Swan
Vaughn Vernon

**Publisher**
Elizabeth Oakley
**Production**
Ralph Bentley
**Illustration**
Jim Carpenter
**Advertising**
Robert Tappan
Andrea McHone
**Subscriptions**
Gary Hoyer
**Business Manager**
Bobbi Sinyard
**Administrative Assistant**
Lori Fanslow

**Founder**
Greg Estes

**Business and Editorial Offices**
Programmer's Journal
150 North 4th Street
Springfield, OR 97477
(503) 747-0800

**Subscriptions**: 1 year (6 issues) $24
($30 in Canada and $40 elsewhere)
**Change of Address**
Programmer's Journal
P.O. Box 30160
Eugene, OR 97403

Articles and letters in Programmer's Journal represent the views of their authors and not necessarily those of the publisher. Letters to the editor become the property of Programmer's Journal and may be edited and published.

Code and listings in Programmer's Journal appear 'as is,' and Programmer's Journal makes no guarantee or warranty of their quality, performance, accuracy, or suitability for any particular purpose. Programmer's Journal assumes no responsibility or liability to any person or entity with respect to any damage caused or alleged to be caused by their use.

Advertising: For ad rates, deadlines, and placement, call Robert Tappan or Andrea McHone (503/747-0800), or write to Programmer's Journal, P. O. Box 30160, Eugene, OR 97403.

Programmer's Journal is produced on Ventura Publisher, a product of Xerox Corporation.

Programmer's Journal (ISSN 0747-5861) is published bimonthly by Oakley Publishing Company. Second-class postage paid at Springfield, OR, and additional mailing offices.

POSTMASTER: Send address changes to Programmer's Journal, P.O. Box 30160, Eugene, OR 97403.

© 1988 OAKLEY PUBLISHING COMPANY

# NOTES from the Editors

Just when my mind was running on bibliographies, I picked up the July *Computer Language* and came upon P.J. Plauger's bibliographical essay. I enjoy a well-wrought bibliography, the more idiosyncratic and critical, the better. What greater pleasure than rambling talk over the good books you've read (in Plauger's case, "and written"), with just enough point to leave room for a little latitude (in Plauger's case, the generous "bibliography for designers and coders of computer programs")? A good bibliography is just plain good reading. A good bibliography . . . I'm getting carried away.

My mind runs on bibliographies because we're a little more academic than usual in Volume 6.5. The bibliographies in this issue are more to a point than Plauger's (though they have some books in common with his), the point coming from a collection of articles huddled under the rubric "new languages," or "languages coming into their own."

The Defense Department and a host of other government agencies and some academicians have settled on Ada, and Richard Riehle thinks Ada's a good choice for any project that calls for a team of programmers and maintenance by someone other than the original coders. Vaughn Vernon's a C programmer, and when it comes to coding for large, complex operating systems such as OS/2, C++ seems to him like the logical extension. Andrew Chalk has been following the fortunes of M.F. Cowlishaw's REXX, an internal success at IBM that's finally emerged as their standard "scripting language" from PC to mainframe, and he suggests that REXX on the PC offers MS-DOS programmers some immediate relief from the inadequacies of the DOS batch facility.

Our traditional preoccupations and languages are represented, too. We were flattered on Michael Abrash's behalf when we received an emergency order from Europe for everything Michael's written on the EGA and the VGA plus code disks. In this issue's installment Michael explains VGA text paging to the tune of a grand total of 32 text pages and provides two assembler listings that should take care of most of your text paging needs. Tom Swan's figured out the details of using dynamic arrays and records in Turbo Pascal 4.0 to save memory and achieve maximum flexibility. Rex Jaeschke checks WATCOM C V6.0 against the ANSI Draft Standard, and W.S. Hall follows up last issue's Windows template with a Presentation Manager template in C. Paul Barkley helps you past two of the consultant's worst dilemmas (assuming you've landed any work at all). We lead off with the redoubtable Hal Hardenbergh, and we close with Steve Baker's experienced account of some everyday programmer's tools—editors, debuggers, and a translator.—*Erin*

I'm writing in the heat of a dry summer what you'll read in the cool of September. The summer doldrums seem to have set in, but some interesting new products will show up in late summer and early fall. 386 Ada and APL compilers will join the Pascal, FORTRAN, COBOL, PL/I, and C compilers already available under the 386 DOS extenders. Borland will soon release Turbo C 2.0 and Pascal with their long awaited source code debugger. High resolution graphics boards sporting the TI 34010 are invading the PC market in ever increasing numbers. And more than ever, the fate of OS/2 as the succcessor to DOS is in doubt.

Assuredly, OS/2 will achieve some success in the marketplace, but its anticipated dominance appears unlikely. Let's look back to the introduction of the IBM PC. Microsoft (PC DOS 1.0 at $50) and Digital Research (CP/M-86 at $300) were selling similar operating systems for the PC. Because of pricing (and DOS had a version of BASIC and faster disk I/O), DOS was bundled with most of the machines sold. OS/2 is really the first PC operating system that has to be "sold" to users. What I'm seeing in the market lately is an operating system squeeze with OS/2 wedged in the middle. At the bottom end is DOS 3.x, and now DOS 4.0 will ship with built-in LIM 4.0 support and a new user shell. OS/2 provides three major enhancements over DOS—a larger address space, multitasking, and virtual memory management. The Presentation Manager adds a Mac-like windowing interface at the price of reduced execution speed. Only OS/2's larger address space is likely to deliver any magnitude of speed improvement in program execution. Sure, if you run disk I/O tests, OS/2 may look better than DOS on the same machine. Build the same disk caching logic into DOS, and this apparent speed advantage slims. IBM may have sold 2 million PS/2s, but over half are low-end models incapable of running OS/2. IBM has to keep those customers happy, too—enter DOS 4.0 and up. And LIM 4.0 provides the potential for a larger address space on older PCs. Now Microsoft Windows/286 at the low end will provide multitasking and a Mac-like user interface, too, albeit at a speed penalty.

High end users are buying 386 machines—Intel has sold about a million chips. At this end of the spectrum we have 386 versions of Unix with VP/IX or Merge/386 DOS emulators that can not only do multitasking and support multiple Unix users, but can also run multiple DOS applications. We have the 386 DOS extenders (from Phar Lap, AI Architects, and IGC) that take advantage of the power of the 386 chips by allowing true 386 applications to be run under DOS. Recent agreements among Quarterdeck (DESQview), Phar Lap, AI Architects, and others mean that DESQview will support multitasking these 386 applications. Phar Lap's DOS extender will soon support virtual memory, and AI Architects is adding support for networking using their Hummingboard 80386/387 coprocessor boards exceuting in parallel. Competing in the middle are the 386 DOS multitaskers—IGC's VM/386, Windows/386, and DESQview's QEMM386. Then there are the PC multitasking/multiuser systems that run DOS—Digital Research's Concurrent DOS-386 and the Software Link's PC/MOS. The bottom line is less and less growing room for OS/2.—*Steve*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# PolyAWK™ – The Toolbox Language™.

## For C, Pascal, Assembly & BASIC Programmers.

We call PolyAWK our "toolbox" language because it is a general-purpose language that can replace a host of specialized tools or programs. You will still use your standard language (C, Pascal, Assembler or other modular language) to develop applications, but you will write your own specialized development tools and programs with this versatile, simple and powerful language. Like thousands of others, you will soon find PolyAWK to be an indispensable part of your toolbox.

### A True Implementation Under MS-DOS

Bell Labs brought the world UNIX and C, and now professional programmers are discovering AWK. AWK was originally developed for UNIX by Alfred Aho, Richard Weinberger & Brian Kernighan of Bell Labs. Now PolyAWK gives MS-DOS programmers a true implementation of this valuable "new" programming tool. PolyAWK fully conforms to the AWK standard as defined by the original authors in their book, *The AWK Programming Language*.

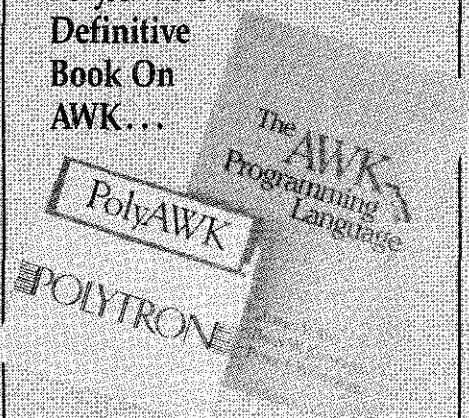
### A Pattern Matching Language

PolyAWK is a powerful pattern matching language for writing short programs to handle common text manipulation and data conversion tasks, multiple input files, dynamic regular expressions, and user-defined functions. A PolyAWK program consists of a sequence of patterns and actions that tell what to look for in the input data and what to do when it's found. PolyAWK searches a set of files for lines matched by any of the patterns. When a matching line is found, the corresponding action is performed. A pattern can select lines by combinations of regular expressions and comparison operations on strings, numbers, fields, variables, and array elements. Actions may perform arbitrary processing on selected lines. The action language looks like C, but there are no declarations, and strings and numbers are built-in data types.

### Saves You Time & Effort

The most compelling reason to use PolyAWK is that you can literally accomplish in a few lines of code what may take pages in C, Pascal or Assembler. Programmers spend a lot of time writing code to perform simple, mechanical data manipulation — changing the format of data, checking its validity, finding items with some property, adding up numbers and printing reports. It is time consuming to have to write a special-purpose program in a standard language like C or Pascal each time such a task comes up. With PolyAWK, you can handle such tasks with very short programs, often only one or two lines long.

### PolyAWK Comes With The Definitive Book On AWK...




Requires MS-DOS 2.0 or above & 256K RAM. **$99**

When you order PolyAWK you receive a copy of *The AWK Programming Language* written by the authors of the original UNIX-based AWK. The book begins with a tutorial that shows how easy AWK is to use, followed by a comprehensive manual. Because PolyAWK is a complete implementation of AWK as defined by the book's authors, you will use this book as the manual for PolyAWK.

You can purchase PolyAWK and the book, *The AWK Programming Language*, for $99. If you already have the book, you can order PolyAWK software only for $85, which is $14 off the regular $99 purchase price. (The book serves as the User's Manual, so you should already have a copy of the book if you are ordering the software only.)

### PolyShell Bonus!

PolyShell gives you 57 of the most useful UNIX commands and utilities under MS-DOS in less than 20K. You can still use MS-DOS commands at any time and exit or restart PolyShell without rebooting. MS-DOS programmers — discover what you have been missing! UNIX programmers — switch to MS-DOS painlessly! PolyShell and PolyAWK are each $99 when ordered separately. Save $50 by ordering the PolyShell + PolyAWK combination package for $149. *Not copy-protected.*

### 30-Day Money Back Guarantee

Credit Card Orders:

**1-800-547-4000**

Ask for Dept. PJL

Send Checks and P.O.s To:
POLYTRON Corporation
1700 NW 167th Place, Beaverton, OR 97006
(503) 645-1150 — FAX: (503) 645-4576

### Prototype With PolyAWK, Translate To Another Language

The brevity of expression and convenience of operations make PolyAWK valuable for prototyping even large-sized programs. You start with a few lines, then refine the program, experimenting with designs by trying alternatives until you get the desired result. Since programs are short, it's easy to get started and easy to start over when experience suggests a different direction. PolyAWK has even been used for software engineering courses because it's possible to experiment with designs much more readily than with larger languages. It's straightforward to translate a PolyAWK program into another language once the design is right.

### Very Concise Code

Where program development time is more important than run time, AWK is hard to beat. These AWK characteristics let you write short and concise programs:

- The implicit input loop and the pattern-action paradigm simplify and often entirely eliminate control flow.
- Field splitting parses the most common forms of input, while numbers and strings and the coercions between them handle the most common data types.
- Associate arrays use ordinary strings as the index in the array and offer an easy way to implement a single-key database.
- Regular expressions are a uniform notation for describing patterns of test.
- Default initialization and the absence of declarations shorten programs.

### Large Model Implementation

PolyAWK is a large model implementation and can use all of available memory to run big programs or read files greater than 64K.

### Math Support

PolyAWK also includes extensive support for math functions such as strings, integers, floating point numbers and transcendental functions (sin, log, etc.) for scientific applications. Conversion between these types is automatic and always optimized for speed without compromising accuracy.

Number 26 on the Reader Service Card

®

POLYTRON

*High Quality Software Since 1982*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

### It Can Be Done

In *PJ* 6.4 Mark Brand asked about the feasibility of sending data from a Wang Minicomputer's printer port to the printer port of a PC. I'm not sure why anyone would want to do this, but it can be done.

On a PC's printer port, there are five (unlatched) input lines. In the normal configuration, they represent printer status; for example, BUSY, OutOfPaper, etc. They can, however, be used to input data, four bits at a time. The fifth line is used as a strobe line to synchronize the data transfer. By choosing pin 10 as the strobe line, you can even set up to generate an interrupt as each nybble comes across the cable, thereby allowing your data transfer routine to run as a background task.

Figure 1 shows the pinout of the required cable. In addition, the handshake lines on the Wang side must be "jumpered" to appropriate states if the mini's standard PRINT A CHARACTER routine is to be used. If you write your own output routine, this is not necessary.

Figure 2 shows the software pseudocode for each side of the transfer. Since only four bits are transferred at a time, the effective throughput of the Wang to PC connection is approximately half of a normal Centronics port, but who really cares?

I doubt that this solution is worth the effort as a printer link, but it certainly demonstrates the danger of saying "it can't be done" to a techno-weenie. For further information, refer to the *IBM Personal Computer XT Technical Reference*, pages 1–111, A-44, and D-25.

*Gordon Arbeitman*
*Gaithersburg, MD*

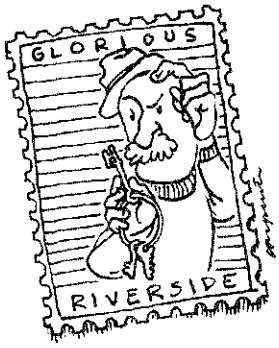



**Figure 1**

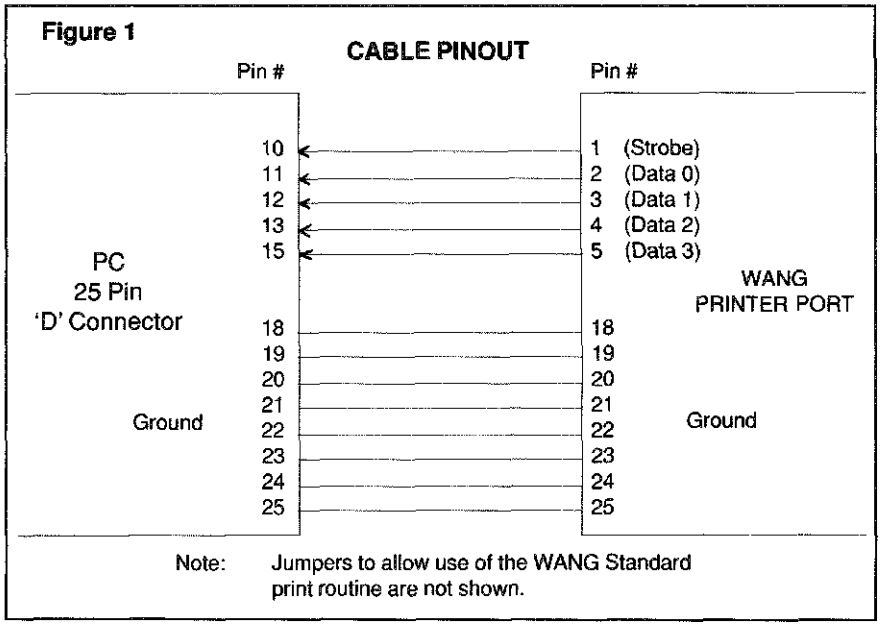



**Figure 2**

```
                                SOFTWARE PSEUDOCODE

                                      PC SIDE

EQUATES

PrintPort = $0379  /* for printer card */
PrintPort = $03BD  /* for mono card */
EOF       = $FFFF  /* End of file marker .. must be  128 */

Procedure WaitForStrobe ;

   Repeat
       Strobe := IN (PrintPort) AND $40 ; /* read bit 6 -> pin 10 */
   until Strobe = 1 ;

   Repeat
       Strobe := IN (PrintPort) AND $40 ; /* read bit 6 -> pin
          10 */
until Strobe = 0 ;

End ;

Repeat

    WaitForStrobe ;
    Inbyte := IN (PrintPort) AND $B8 ; /* Get the low nybble */
    WaitForStrobe ;
    Temp := ShiftLeft (IN (PrintPort) AND $B8), 4 ; /* Get the
      high nybble */
    InByte := InByte OR Temp ;
    Store InByte .. or send it to the printer or whatever ;
Until (InByte = EOF) ;

END

                                  MINICOMPUTER SIDE

Repeat for each OdtByte ;

    Print OutByte ;                    /* send the low nybble */
    ShiftRight OutByte, 4 ;
    Print OutByte ;                    /* send the high nybble */

Until (NoMoreCharacters) ;

Print EOF ;
END.
```

*The method described above works on an unmodified PC parallel printer port, but the scheme requires software at both ends. I recently spoke with Fred Cisin of Xenosoft (Berkeley, California), who is working on a new product to attack this problem. (Xenosoft specializes in the conversion of data among various machines and disk formats.) Just a single cut and the addition of a jumper wire will make many parallel printer cards bi-directional. Fred is working on a modification to inexpensive Taiwanese parallel port cards and PC software that would allow the PC's parallel port to be used for input and output. The advantage is that software is required only on the PC, and this would allow any kind of computer with a parallel port to transfer data to a PC with such a modified parallel port and appropriate software. The other attractive benefit is the potential transfer speed, much faster than with serial interfaces.—MSB*

### F11 and F12

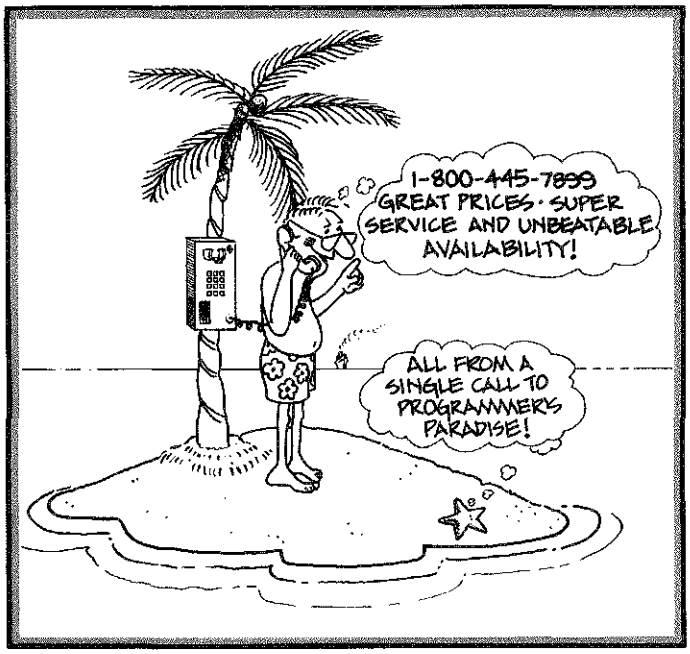
I am developing several data entry screens using Borland's Turbo C 1.5. One requirement is that function keys F11 and F12 be used; however, **getc()** or a BIOS call, **16h**, reveal nothing. I've found little documenta-

*Continued . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# PARADISE PRICES 1-800-445-7899




CALL PROGRAMMERS PARADISE TODAY **and discover our commitment to bringing you the best software at the best prices. You'll also find software pros to help you select the product that you need plus immediate shipment on our stock of 1000 products and a 30-day money back guarantee. Paradise is ready and waiting for you.**

### 386 SOFTWARE

| | |
|---|---|
| 386-to-the-Max | 66 |
| FoxBASE+/386 | 459 |
| Microport System V/386 (comp.) | 759 |
| MS Windows/386 | 130 |
| NDP C or Fortran-386 | 529 |
| Pharlap 386\|ASM/Link | 409 |
| SCO Xenix System V 386 (comp.) | 1279 |
| VM/386 | 182 |

### ASSEMBLERS/LINKERS

| | |
|---|---|
| Advantage Disassembler | 279 |
| MS Macro Assembler | 99 |
| Optasm | 109 |
| Plink86plus | 279 |

### BASIC

| | |
|---|---|
| Flash-Up | 80 |
| MS Basic Compiler 6.0 | 189 |
| MS QuickBASIC | 69 |
| QuickPak | 60 |
| Turbo Basic | 69 |
| Turbo Basic Toolboxes | 69 |

### C COMPILERS/INTERPRETERS

| | |
|---|---|
| C-Terp | 232 |
| Lattice C | 272 |
| Microsoft C | 289 |
| Quick C | 69 |
| Run/C Professional | 159 |
| Turbo C | 69 |

### C LIBRARIES

| | |
|---|---|
| C Asynch Manager | 137 |
| C Tools Plus/5.0 | 101 |
| C Utility Library | 125 |
| Essential Comm Library | 125 |
| Communications Plus | 199 |
| Greenleaf Comm Library | 169 |
| Greenleaf Functions | 155 |
| PforCe | 215 |
| *Timeslicer* | 279 |
| Turbo C Tools | 101 |

### COBOL

| | |
|---|---|
| Micro Focus COBOL/2 | 733 |
| Microsoft COBOL | 452 |
| Opt-Tech Sort | 105 |
| Realia COBOL | 794 |
| Realcics | 799 |
| SCREENIO | 382 |

### DBASE TOOLS

| | |
|---|---|
| Clipper | 399 |
| Eagle | 395 |
| FoxBASE+ | 269 |
| Hi-Screen XL | 129 |
| R & R | 139 |
| Say What?! | 45 |

### DEBUGGERS

| | |
|---|---|
| Periscope I | 373 |
| Periscope II | 141 |
| Pfix 86 Plus | 215 |

### DISK/DOS/KEYBOARD UTILITIES

| | |
|---|---|
| Command Plus V. 2.0 | 70 |
| Disk Optimizer | 56 |
| Fastback Plus | 142 |
| Norton Commander | 56 |
| Norton Utilities Advanced | 99 |
| PC Tools Deluxe | 70 |

### TROPICAL TREATS

Greenleaf has added several new products to its outstanding line of development tools including a 200 functions mathematics library and a new interactive screen generator/editor for Greenleaf DataWindows. Call us to order any of these or other Greenleaf Software Products.

| | |
|---|---|
| Greenleaf Functions | 155 |
| Greenleaf Comm. Library | 169 |
| Greenleaf DataWindows | 209 |
| DataWindows for OS/2 | 279 |
| TurboFunctions | 109 |
| Business Mathlib *NEW* | 239 |
| SuperFunctions *NEW* | 265 |
| Makeform *NEW* | 125 |

### EDITORS

| | |
|---|---|
| Brief | CALL |
| w/dBrief | CALL |
| EDIX | 169 |
| Epsilon | 151 |
| KEDIT | 120 |
| Norton Editor | 70 |
| PC/EDT+ | 269 |
| SPF/PC | 185 |

### FILE MANAGEMENT

| | |
|---|---|
| Btrieve | 185 |
| Xtrieve | 189 |
| Report Option | 109 |
| Cbtree | 141 |
| c-tree | 318 |
| r-tree | 241 |
| dBC III | 172 |
| dB_Vista or dB_Query | CALL |
| Informix Products | CALL |

### FORTRAN

| | |
|---|---|
| Lahey Personal FORTRAN *77* | 86 |
| MS FORTRAN | 289 |
| RM/FORTRAN | 479 |

### GRAPHICS

| | |
|---|---|
| Essential Graphics | 229 |
| HALO '88 | 229 |
| HALO '88 (5 Microsoft Lang.) | 399 |
| MetaWINDOW Plus | 232 |

### OBJECT-ORIENTED PROGRAMMING

| | |
|---|---|
| Actor | 423 |
| Advantage C++ | 479 |
| PforCe++ | 215 |
| Smalltalk/V | 85 |
| Application Packs | 45 |
| Smalltalk/V286 | 169 |

### OPERATING SYSTEMS

| | |
|---|---|
| Microport 286 DOS Merge | 129 |
| Microport Sys V/AT | 469 |
| *SCO Xenix System V (comp.)* | 999 |
| Wendin-DOS | 80 |

### 1000 BRAND NAME PRODUCTS IN STOCK

### PASCAL COMPILERS

| | |
|---|---|
| Microsoft Pascal | 189 |
| Turbo Pascal | 69 |
| Turbo Pascal Dev. Lib. | 289 |

### TURBO PASCAL ADD-ONS

| | |
|---|---|
| T-Debug Plus | 39 |
| w/Source | 80 |
| Turbo Asynch Plus | 101 |
| Turbo Halo | 80 |
| Turbo Magic | 90 |
| Turbo Pascal Toolboxes | 69 |
| Turbo Professional 4.0 | 80 |

### SCREENS/WINDOWS

| | |
|---|---|
| Greenleaf DataWindows | 209 |
| Microsoft Windows | 69 |
| MS Windows Development Kit | 319 |
| Panel Plus | 395 |
| Turbo Power Screen | 101 |
| Vitamin C | 162 |
| VC Screen | 119 |
| Windows for Data | 259 |

### ADDITIONAL PRODUCTS

| | |
|---|---|
| Carbon Copy Plus | 142 |
| Dan Bricklin's Demo Prog. II | 179 |
| Eureka | 119 |
| FLOW CHARTING II | 205 |
| MathCad | 279 |
| MS OS/2 Prog. Toolkit | 239 |
| Norton Guides | 69 |
| PC-Lint | 101 |
| PC Scheme | 86 |
| Polytron PVCS | CALL |
| Pre-C | 159 |
| Source Print | 80 |
| Tree Diagrammer | 70 |

## HOW WE WORK

### PHONE ORDERS

Hours 9 AM–7 PM EST. We accept MasterCard, Visa, American Express. Include $3.95 per item for shipping and handling. All shipments by UPS ground. Rush service available, ask for rates when you order.

### MAIL ORDERS

POs by mail or fax are welcome. Please include phone number.

### TECHNICAL SUPPORT FROM SOFTWARE PROS

### RETURN POLICY

Our "No Hassle" policy means if you're not satisfied with a product from Programmer's, simply return it within 30 days for a refund. Some manufacturer's products cannot be returned once disk seals are broken, so check before you buy.

### INTERNATIONAL SERVICE

Take advantage of our International business number for details on export charges and exchange rates. Payments should be made in U.S. dollars.

### DEALERS AND CORPORATE ACCOUNTS

Call and ask for our catalog and special volume discounts.

### UNBEATABLE PRICES

We'll match lower nationally advertised prices.

In NY: **914-332-4548**
Customer Service: **914-332-0869**
International Orders: **914-332-4548**
Telex: **510-601-7602**
Fax: **914-332-4021**

**Call or Write for Latest Free Catalog!**

# 1-800-445-7899

Programmer's Paradise™

**A Division of Magellan Software Corp**
55 South Broadway, Tarrytown, NY 10591

Number 200 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If you can find better sort/merge software, buy it!

**Sortex:®** A flexible, general purpose sort/merge facility which provides you with the ultimate in performance and reliability

- Runs from the DOS prompt, batch file, or applications written in Assembler, BASIC, COBOL, FORTRAN, Pascal, or C.
- Accepts your requirements through the full screen menu/panel facility, control file, or command line.
- Processes most of the commonly used file and data types.
- Produces output files with key, address, index, tag, or full records.
- Imposes no practical limitations. Files with four billion characters, records & fields with thirty two thousand characters each, and unlimited number of keys can be handled by Sortex.
- Allows records to be chosen selectively for processing.
- Sorts your records in ascending/descending order on any number of keys.
- Merges your sorted record sequences into one sequence on any number of keys.
- Preserves the input order of records on the output file if all their keys are identical.
- Allows keys to overlap, or be contained within other keys. They need not be contiguous.
- Processes files that contain both standard and nonstandard delimiters.

30-Day Money-Back Guarantee
Only ~~$149.95~~ $49.95

SYNCSORT
CORPORATION
1915 W. Orangewood Avenue
Orange, California 92668
Tel.: 714 978 0480

Number 171 on the Reader Service Card

tion regarding these two function keys. One book, *DOS Customized*, states that F11 and F12 have scan code values of 133 and 134, respectively, but says nothing about how to use these function keys. I've even tried using the ANSI.SYS driver, with no luck. After all of this, I received a copy of Word Perfect 5.0 that successfully uses F11 and F12.

My question: How do you utilize these keys? I'd appreciate any light you can shed on this matter.

*John P. Brannigan*
*Riverside, CA*

*Over the last five years, IBM has produced three different keyboards. The most recent, the 101-key Enhanced Keyboard, is now considered the new standard. However, it is not supported on most machines manufactured before 1987. Most ROM BIOSes before 1987 will not recognize the F11 and F12 keystrokes. In even the newer machines, the BIOS interrupt 16h read keyboard (AH=0) call will discard the F11 and F12 keystrokes. The BIOS interrupt 16h extended keyboard read function (AH=10h) must be used to retrieve a keyboard buffer code. F11 returns the code 8500h, and F12 returns 8600h.*

*Another consideration is that many enhanced, twelve-function keyboards treat F11 and F12 merely as the equivalents of Alt-F1 and Alt-F2, respectively. Other enhanced keyboards have switches or jumpers that allow you to configure the F11 and F12 keys as independent function keys if your ROM BIOS fully supports the new enhanced keyboard.*

*An article on the enhanced keyboard appears in* PC Tech Journal, *July 1987. In "Keying on a Standard" Bob Smith describes the different scan-codes, discusses some of the programming problems, and provides some routines to determine the keyboard type and provide information about individual keys of a selected keyboard type.—Ed.*

### The PS/2 Revisited

In the 6.3 issue, I raised doubts about reported PS/2 sales, doubts based on my making only two shipments out of 100 on 3.5-inch disks and receiving only one additional inquiry about the availability of my product on 3.5-inch disks. The recent four months through the end of June 1988, have seen one 3.5-inch disk in over 60 shipments and no additional inquiries. This recent data does not duplicate my previous report. It shows the PS/2 at no more than a 2% penetration rate, a downward revision from my previous 2–3%. In 6.4, Chuck Gilmore replied with a defense of OS/2 and PS/2. He started by noting that he, too, has had only two requests for the 3.5-inch medium but does not give the corresponding 5.25-inch figures. Thus, I take it that 2% penetration still stands as a reasonable estimate.

If IBM's claims of several months ago that they had shipped one million PS/2s were true, 2% penetration would imply that at least 50 million PC compatibles were installed worldwide (we have sold into more than 15 countries). This does not reconcile with my estimate of 15 million installed PC compatibles and is the basis for my original call for others to share their data.

One of our machines is a PC compatible with a 3.5-inch disk drive mounted internally using the normal disk controller and PC power supply. I think this is a more elegant solution to the 3.5-inch disk problem than the external drives or machine-machine cables mentioned in Mr. Gilmore's letter. It is also less costly because I did not buy a PS/2.

That IBM is heavily invested in PS/2 is no more compelling than their previous commitment to the RT. Buyers make the ultimate choice. Mr. Gilmore invokes the early history of the PC as a reason to bet with IBM. Recent history has been far more equivocal (PCjr, XT 286, Convertible, and AT hard disks). I have not seen buyer-oriented justification for the microchannel bus. It may be solving IBM's problems, but I do not think it solves mine.

As far as OS/2 is concerned, let me repeat an observation I have been making privately since the late '60s: "The function of an operating systems is to sell memory." Given the current high memory prices, I doubt that a memory hog such as OS/2 will sweep through the market quickly. Half the PS/2s sold can't run OS/2. I find it easy to decouple the fortunes of PS/2 and OS/2. My original letter noted that we'd received no inquiries about OS/2 versions of our product, and the current figure including the last four months is still zero. IBM is pursuing a high-risk strategy. They did not consult with me prior to engaging in it, and I have no obligation to assume their business risks. More than a year after the introduction of the PS/2, it represents a small percentage of the available market, one which is adequately served via an add-on to the existing system. Indeed, most ads in *Programmer's Journal* don't bother to mention either the PS/2 or the 3.5-inch medium.

*Dennis Reinhardt*
*DAIR Computer Systems*
*Palo Alto, CA*

### Chuck Gilmore Replies

I still take my stand with PS/2 and OS/2. I'd draw an analogy from the automotive industry—most cars on the road today were manufactured before 1987 (I choose 1987 because that's the year IBM introduced the PS/2 series). This does *not* in any way imply that people are not buying 1987 or 1988 cars (or 1987 or 1988 PS/2 machines). That 1987 and 1988 car models are so new and previous models have been around so much longer suggests that the percentage of new models among the old is bound to be low (I would venture to guess 5% or less).

That your company shows a 2% rate of sales in the 3.5-inch disk medium does not



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# VEDIT PLUS

# #1 PROGRAMMABLE EDITOR

## Call 1-800-45-VEDIT for FREE Fully Functional Demo Disk

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS easily satisfies the most demanding computer professional.

### Try a Dazzling Demo Yourself.

The free demo disk is fully functional—you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial—you experiment in one window while another gives instructions.

The powerful "macro" programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a "macro"—it shows that no other editor's "macro" language even comes close. And VEDIT PLUS is only 40K in size.

Go ahead. Call for your free demo today, You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

### Only VEDIT PLUS is this Flexible.

The installation lets you pick from closely emulating the keyboard layout of Word Perfect, WordStar and others. Or you can easily create your own layout and even your own editing functions. Supports any screen size—you pick screen colors and attributes.

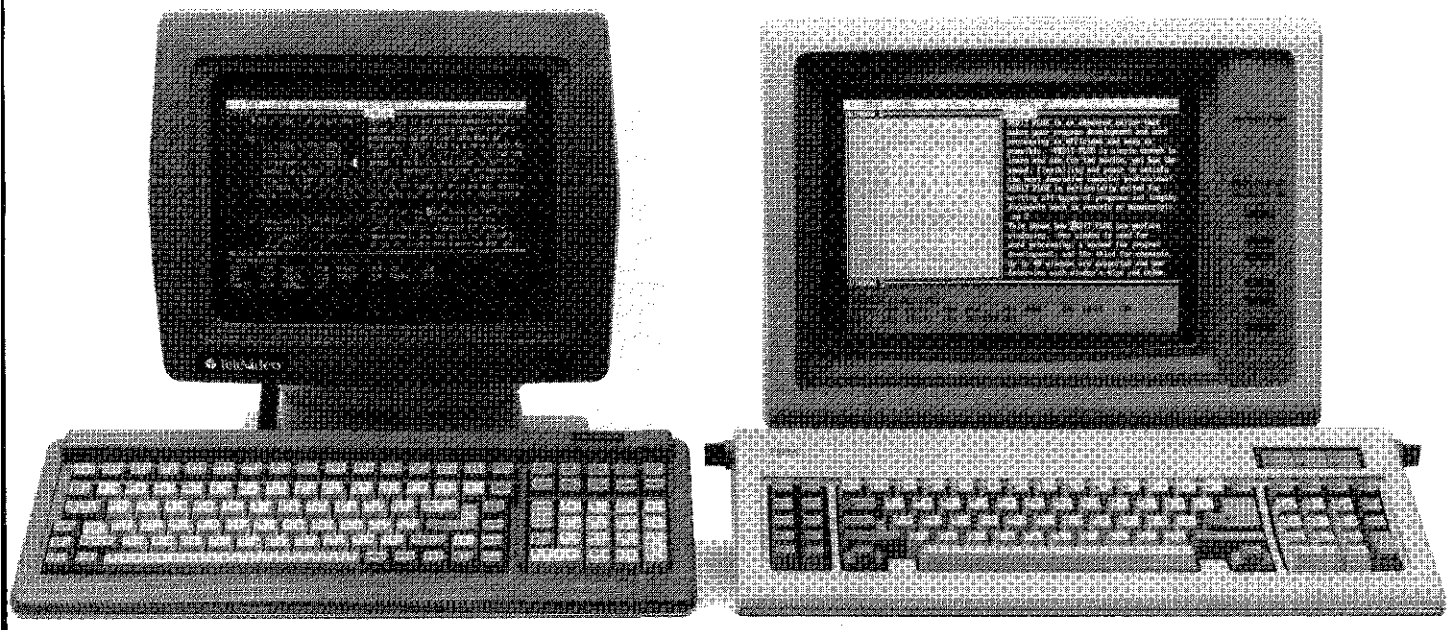
Supports the IBM PC, XT, AT and PS/2. Also supports MultiLink, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, FlexOS (protected mode), CP/M-86 and CP/M. (Yes, we support windows on most CRT terminals, including CRTs connected to an IBM PC.) Order direct or from your dealer. $185.

**Special:** VEDIT (single file, no windows) for CP/M—$49.

- **Fully Network Compatible**
- **Call for XENIX-286 version**
- **30 Day Money-back guarantee**

### Compare Features and Speed

| | BRIEF | Norton Editor | PMATE | VEDIT PLUS |
|---|---|---|---|---|
| 'Off the cuff' macros | No | No | Yes | Yes |
| Built-in macros | Yes | No | Yes | Yes |
| Keystroke macros | Only 1 | No | No | Unlimited |
| Multiple file editing | 20+ | 2 | No | 20+ |
| Windows | 20+ | 2 | No | 20+ |
| Macro execution window | No | No | No | Yes |
| Pop-up menus | No | No | No | Yes |
| Execute DOS commands | Yes | Yes | Yes | Yes |
| Automatic processing of Compiler errors | Yes | No | No | Yes |
| "Cut and paste" buffers | 1 | 1 | 1 | 36 |
| Undo line changes | Yes | No | No | Yes |
| Paragraph justification | No | No | No | Yes |
| Convert to/from WordStar | No | No | No | Yes |
| On-line calculator | No | No | No | Yes |
| Configurable Keyboard | Hard | No | Hard | Easy |
| 43 line EGA support | Yes | No | No | Yes |
| Manual size/index | 250/No | 42/no | 469/Yes | 380/Yes |
| Benchmarks in 120K File: | | | | |
| 2000 replacements | 1:15 min | 34 sec | 1:07 min | 6 sec |
| Pattern matching search | 20 sec | Cannot | Cannot | 2 sec |
| Pattern matching replace | 2:40 min | Cannot | Cannot | 11 sec |



VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc. MultiLink and PC-MOS/386 are trademarks of The Software Link, Inc. CP/M and FlexOS are trademarks of Digital Research. MS-DOS is a trademark of Microsoft.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, Wyse WY700 and others.
*Demo disk is fully functional, but does not readily write large files.

# CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299, TELEX 701821



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

take into account the type of product you sell or the audience it caters to. I would not make the assumption that my low percentage of 3.5-inch disk requests is an accurate indicator, either, since my product doesn't cater to each and every person or company who has a computer. I would assume that your product is targeted at a specific group of people. Your 2% penetration rate for the 3.5-inch medium may be accurate for your type of product, but to carry that figure over into the entire community of personal computer users is not justified.

If you'll recall, the entire theme of my last letter (*PJ* 6.4) was that users of PS/2s don't always request the 3.5-inch medium because the more popular (at this time) 5.25-inch medium can be read on older machines as well as on PS/2s equipped with external 5.25-inch drives, or transferred with "bridge" programs from 5.25-inch based machines to PS/2s. Those opting for one of these two routes would further throw off your percentage. The fact that you've installed a 3.5-inch drive on your PC compatible may solve *your* problem of producing 3.5-inch copies of your software, but it does not address the point.

You are correct in that most of the ads appearing in *Programmer's Journal* do not mention PS/2 or OS/2. In the previous issue (6.4), 6 out of the 78 advertisers in the advertiser's index mention OS/2 (7.69%), and 12 ads mention PS/2 (15.38%). Actual percentages are even higher because advertisers selling books, T-shirts, and magazines do not belong in the overall formula. This is definitely not *most* ads, but what was the percentage last year or even 6 or 3 months ago?

You mention IBM's RT computer. Let's be fair—the RT was never meant to be offered to the typical computer user. The RT was released as a computer catering to the scientific community only, *not* to general purpose microcomputer users. As Hewlett Packard seems to have a monopoly in the scientific community, yes, the RT was unsuccessful. The PCjr, also a flop, was never meant to appeal to the general purpose computer user either. The PCjr was to appeal to kids, something that other manufacturers already have a monopoly on. So let's stick to the standard PC, XT, AT, and PS/2 series of computers.

As far as buyer-oriented justification for the microchannel bus is concerned, there is nothing available for the microchannel bus *yet* (that I know of), which is why there's no buyer-oriented justification. I can understand the lack of excitement among those who have never worked with mainframes in a "systems oriented" capacity, or among those who don't understand the meaning and power of the word "channel." Channels are like having separate computers controlling your disk drives or other peripherals, enabling the main CPU to go about doing more important things than wasting its time waiting for a read or write operation to take place before continuing. Eliminating the need for software to perform these tasks increases overall system performance. If no buyer-oriented justification is anticipated, how come Tandy and Dell Corp. will be among the first to produce PS/2 clones incorporating microchannel architecture? If you've been following these two companies, you know they don't market unpopular machines!

I can't help but disagree with your observation about operating systems and OS/2 in particular: "The function of an operating system is to sell memory." If that's the case, why are so many DOS based users buying extended memory for their PCs? Why is it that companies are always complaining about insufficient memory? Since Microsoft is not in the hardware business, what would they have to gain by selling operating systems? Recall that Digital Research (CP/M) and SCO (Xenix) don't sell hardware either. As for OS/2 itself, I'm looking forward to running a BBS in the background while I'm downloading a file from CompuServe with yet another communications program, while my favorite accounting program is printing out the month's revenue and my "make" utility is compiling and linking a programming project, while I type up a letter—all without having to worry about whether some program will hang my machine by writing to memory locations it's not supposed to, or re-vector interrupts that other programs depend on, or trap keys that other programs use. I'll gladly spend the extra bucks for more memory to be able to do this. It's cheaper than running 5 separate tasks on 5 separate computers (not to mention the cost of a LAN if you want them to share the same data). If it's any consolation, IBM did not consult with me either about pursuing PS/2 machines or the OS/2 operating system!

*Chuck Gilmore*
*Gilmore Systems*
*Beverly Hills, CA*

**Virus**

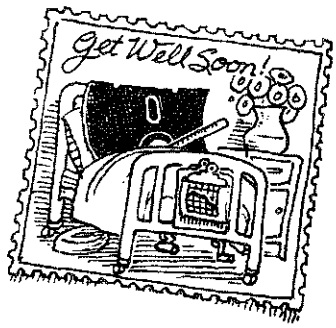
My machine was recently infected with a virus. The damage resulting from the infection took three months to repair. The symptoms started when I was unable to write to the disks because they were reported as full by the operating system. This didn't seem logical since some of the disks had small data files on them. I started using the backup disks, but they also became infected and damaged beyond repair. After weeks of trial and error, neither I nor my friends were able to discover what caused the problem. As a result of their helping me, my friends' systems also became infected via the disks used in my system. I don't know how far the virus spread beyond my immediate group.

I frequently read your magazine and haven't seen any articles forewarning your readership of software viruses. I eventually read a *Newsweek* article that described the problem as a software virus. It was astounding to find the virus problem discussed in a magazine covering general news events when the two computer magzines I read didn't mention them. I've subsequently read about viruses in both the *Washington Post* and *Time*. The *Post* dedicated all of page three in the Sunday edition to the growing problem.

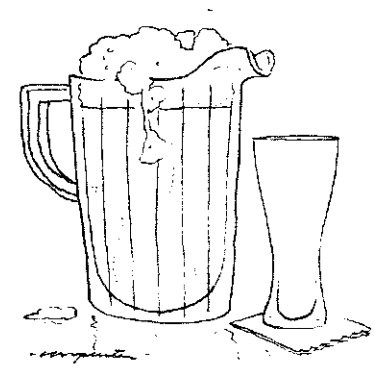
I think it's your responsibility to keep your readers informed about computer problems that could result in months of work's being lost. The damage, just to my system, could buy subscriptions to ten computer magazines for the next five years.

I'm certain that many of your readers are as ignorant of the problem as I was, and the problem is certainly widespread enough to draw national media coverage. It would be a disservice to your readers not to give them information about this problem. Keep us informed of software viruses, symptoms, and cures.

*Ralph Allen*
*Arlington, VA*

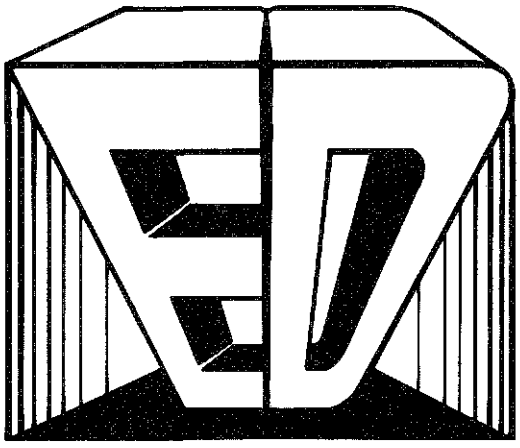



*We agree that viruses are an inimical plague to programmers and users alike. That's why we published a strong letter about them from Gerry Power in our May/June 1988 (Volume 6.3) issue. As the operator of a bulletin board (Turbo Source Search out of Scituate, Massachusetts), Gerry has particular reason to be concerned about the havoc viruses can wreak. In his letter Gerry described how viruses threaten systems, the not always reliable symptoms of infection, preventive measures (neither are they always reliable), and what programmers to date have been able to do about viruses.—Ed.* ♦



— MEMORY EDITOR—


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




TM

# EDITING SYSTEM

## Solidity, Speed, and Power.

ED is an object-oriented open architecture system for editing programs and manuscripts and manipulating data files. Flexibility, extensibility, and programmability are realized by providing access to the objects which ED itself manipulates. All aspects of ED's appearance and operation can be controlled by the programmer. ED, a restricted or enhanced form of ED, or any of ED's objects, can be embedded in the programmer's applications with no royalties. Objects such as dynamic arrays, windows, data entry windows, macros, menus, browsers, popup directories, and regular expressions are manipulated through normal C function calls. Functions for creating manuscripts are part of ED's design. Stream blocks, word-wrap, tabs, intelligent paragraph formatting, block justification, and pagination behave correctly and execute instantly. Finally, ED can be used to create sophisticated sorts and filters. Macros, search and replace, column blocks, and block sorts allow records to be sorted and selectively removed, and fields to be added, rearranged, and removed.

**ED** • NUMBER OF BUFFERS LIMITED ONLY BY AVAILABLE RAM • NUMBER OF WINDOWS LIMITED ONLY BY SCREEN SIZE • **POPUP DIRECTORY FACILITY** • COPY • RENAME • DELETE • BROWSE • FILE STATUS • EDIT • EXECUTE • SORT • **SEARCH AND REPLACE** • FORWARD/BACKWARD • CASE SENSITIVE/INSESITIVE • FULLY PARENTHESIZED REGULAR EXPRESSIONS • INCREMENTAL/GLOBAL REPLACE ACROSS ALL BUFFERS • **BLOCK COMMANDS** • COLUMN/LINE/STREAM BLOCKS • SAVE • CUT • DELETE • BLANK • SEARCH-FOR • FORMAT • JUSTIFY LEFT/RIGHT/CENTER • UPPER/LOWER CASE • REMOVE • OVERLAY • REACTIVATE • TAB • DRAG • SORTS • **MACROS** • MENU DRIVEN • VIEW • AUTO-EXECUTE • TIME-DELAY • NESTED • RECURSIVE • SIZE LIMITED BY AVAILABLE MEMORY • **MORE FUNCTIONS** • VIEW/MANIPULATE BUFFERS THROUGH MENU OR KEYSTROKE • INTELLIGENT DIAGRAM/BOX DRAW • ADJUSTABLE SCROLL VALUE • TRANSPOSE LINES/CHARACTERS/WORDS • MOVE-TO-POSITION STACK • VISUAL TAB CREATION • COLUMNAR/ENTIRE-LINE TABS • TIME AND DATE STAMP • RESTORE TYPED-OVER CHARACTERS • CHANGE DIRECTORY • BRACKET/BRACE/PARENTHESIS MATCHING • ENTER GRAPHICS CHARACTERS • EXCUTE DOS COMMANDS • POPUP ASCII TABLE • USER DEFINED MENUS/POPUP FILES • KEYBOARD CONFIGURATION • PRINT BUFFER WITH PAGE NUMBERS, ADJUSTABLE SPACING AND MARGINS • INTERACTIVE TUTORIAL • ALL DOCUMENTATION ON-LINE • WRITTEN IN C AND ASSEMBLER• **EXTERNAL UTILITIES** • INTELLIGENT CHANGE DIRECTORY COMMAND • POPUP DIRECTORY FACILITY • STRING TRANSLATOR • **HARDWARE REQUIREMENTS** • DOS • PC/AT, PS/2, 386 • CGA, MDA, EGA, HERCULES, WYSE 700/AMDEK 1280 • RUNS IN ALL VIDEO MODES, NO FLAGS, NO DRIVERS • 256K • **FOR LITERATURE, A COMPLETE LIST OF FUNCTIONS, AND A DEMO DISK, CALL 201-450-4545.**

**VERSION 1.0**

**PRICE: $265.00**

THE AMERICAN COSMOTRON 80 HOLMES ST • PO BOX 128 • BELLEVILLE, NJ 07109

VISA MasterCard

Number 215 on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Computers in the Fast Lane

by
**Hal W. Hardenbergh**

*Lord Byron's daughter Ada, the Countess Lovelace, had a friend who was developing a computer. Attempting to secure some venture capital for him, she wrote about how wonderful the new computer would be. That was the first announcement of a wonderful new computer. It was highly exaggerated. Charles Babbage never did get his CPU, which he called an "Analytical Engine," to work.*

Computers have been doubling in speed every two years since the first working computers were developed back in the 1940s. Product *announcements* have been doubling in speed *annually* ever since Ada tried to raise money for Charlie.

**How to Build a Computer:**
To build a computer these days you get an ingot of very pure crystalline semiconductor and you slice it into wafers. You coat the wafer with an optically sensitive substance and expose a pattern onto it. Then you pop the wafer into a specialized oven, much like the big pizza ovens of recent memory. You repeat this, with variations, several times. Then the wafer is cut up into chips that become part of a computer—perhaps the CPU itself. Everybody in the computer business builds their computers this way.

So why are computers doubling in speed every two years? The biggest reason is that the patterns are getting smaller. This has two effects. One is that each piece of the pattern has less capacity (as in resistance and inductance) associated with it. Capacity fundamentally limits computer speed the way that air friction fundamentally limits airplane speed. As the patterns get smaller, the computers get faster.

The other thing that happens is that as patterns get smaller, the individual transistors on the chip get smaller. We can have more transistors on a chip of a given size, so we can make computer parts of greater complexity. More complex computers are faster.

If more complex computers are faster, why don't the computer makers just build larger devices? They do, but there are random unwanted impurities in the semiconductor wafers. The larger the device, the greater the chance that an impurity is going to make the chip inoperable. Since we don't know how to go in there with a hammer and saw to fix those impurities, we have to throw the chip away. Throw too many chips away and the few good ones get expensive. This is why a 386 costs more than the 4-bit microcomputer in your microwave oven. Still, the CPU is almost always the largest chip in the computer because that makes it run faster.

If we can make the crystalline semiconductor ingots with fewer impurities, we can make larger chips for a given yield. This is in fact happening, and it's the third reason why computers are getting faster.

Computers have been doubling in speed every two years for a long time. If you think that's because bright engineers are figuring out better ways to manipulate ones and zeros, you're wrong. In recent years (since the invention of the integrated circuit) the reason is that I.C. patterns are getting twice as dense (a 41% decrease in each linear dimension), *and* the chips are getting 25% bigger, every two years.

***Product* announcements *have been doubling in speed* annually *ever since Ada tried to raise money for Charlie.***

But there are faster computers out there than those based on Intel's and Motorola's latest offerings, aren't there? Like IBM mainframes or Seymour Cray's toys? Right. Motorola, with its newish 68030 and Intel with its 386 make the fastest *mass-market* single-chip CPUs. If it's a mass-market part, you have to be able to make them (literally!) by the millions. (One could quibble over that "single-chip" bit; both need FPUs, and the 386 in particular needs an external cache.)

It must be easy to make a faster CPU than the 386 or 68030 because lots of companies do it every day. Every mainframe or supercomputer maker does, and many of the minicomputer makers do, too. You can even use a handful of gate arrays to emulate a microprocessor and make microprocessor code run a *lot* faster. Edge Computer has been shipping such devices for a year now. But the mass-market seems to be based on what we can loosely call single-chip CPUs.

**How to Announce a Computer:**
You get a piece of paper and, following Countess Lovelace's early example, write down whatever outrageous nonsense crosses your mind. For instance, you might, on the occasion of introducing a computer based on a new single-chip CPU, assert that your company's offering will double in speed *every year*, for the next *ten years!* In fact, Sun Microsystems did exactly that last year when they introduced their SPARC-based machines.

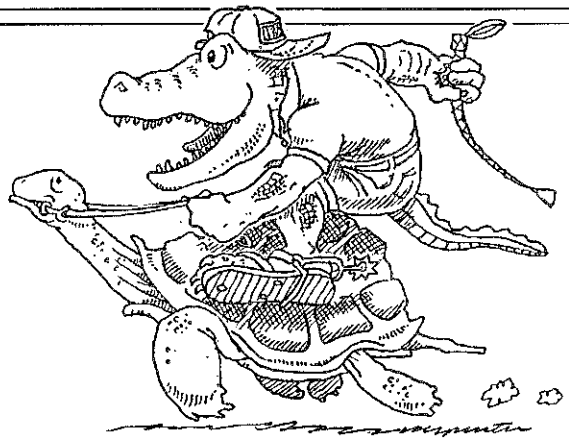
Now, what did Sun mean by that announcement? They could have meant that after a year they will put two SPARCs on the table, then four SPARCs on the table the year after that. Nah. You can go back to 1976 and play that game with 6502s, but in 1988 you'd need a large tabletop to hold 4096 of them. I don't think 4096 6502s are 4096 times as fast as one 6502. A lot of people agree with me.

Perhaps Sun means that it will gradually edge into the mainframe business, where you really *can* run a lot faster than the current SPARC! No, the announcement seemed to be that the SPARC, a single-chip CPU, would double in speed every year.

Perhaps Sun has access to some advance information and knows that the computer industry in general is about to double its historic rate of performance increase, a rate that has been constant for 40 years. No, the announcement implied that it was *Sun* and the *SPARC* that would exhibit this remarkable performance growth! The computer industry in general will show performance increases of only 32X over the next 10 years. Sun is asserting a 1024X increase, which means that they will be outperforming the industry by a factor of 32 in 1997.

To put such a performance advantage in perspective—a very good IBM mainframe computer will, today, outperform the 386 by about 32X. It would seem that Sun will be building SPARC-based machines ten years from now that, as single-chip minicomputers, will perform at the same level as then-current mainframes. Big mainframes. I'd like to congratulate Sun, in advance, for this remarkable achievement.


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



Sun obviously expects to parlay some combination of technology to get this 32X leg up on the industry. Do they know how to make patterns smaller? Can they make purer crystalline semiconductor ingots? (If you believe that Sun has a bright engineer or three who know very clever ways to interconnect registers, accumulators, and stacks to achieve this 32X improvement, you must *also* believe that some Detroit engineer invented a carburetor with which a 1967 Chrysler gunboat gave 257 miles per gallon! Technology is wonderful, but not that wonderful.)

We have a problem here. You see, Sun doesn't make crystalline semiconductor ingots. Or the chemical "resist" that coats the wafers. Or the steppers that replicate the pattern on the wafers. Or the "pizza ovens" to bake the wafers. Neither does any of the other traditional CPU houses, especially (for the audience reading this) Intel and Motorola.

No, everybody in the industry buys ingots from the same vendors. They buy resist from the same chemical firms. They buy steppers and "pizza ovens" from the same companies. And they hire engineers from the same labor pool. *The computer industry is not based on separate enclaves of engineers working in mutual isolation, with the best bunch of engineers winning.* The technology is in the air, and everybody breathes the same air.

Sun Microsystems does not even own its own wafer fab—the SPARC is contracted out to various foundries. Four of them, in fact. The total number of SPARCs those four foundries will turn out in the next year is fewer than the number of 386s Intel will make this morning between coffee break and lunch.

IBM has some proprietary computer hardware technology, but then IBM is the U.S.'s second biggest manufacturer (behind General Motors), and is 50 times bigger than Sun. •

*Sun has no hardware technology whatever, but Sun's hardware is going to outperform the industry by 32X.* Does this make sense?

It does to Sun. You see, they are going to conquer the world—well, the computer industry—with a better arrangement of registers and accumulators and stacks. They are going to throw away all the slower parts of modern CPUs and keep just the faster parts. The result, by 1997, will outperform the industry by a factor of 32X. This wonderful new arrangement is called RISC.

Sun began shipping SPARC-based RISC computers a year ago. They had been making Unix minicomputers based on the 4 MIPS 68020; they announced that their SPARC RISC machine performed at 10 MIPS. Since their new SPARC machine was obviously 2.5 times faster than the old 68020 machine . . . *obviously??*

There are a few curmudgeons and naysayers who think that when the RISC folks throw out the complex stuff in a traditional CPU that they also throw out the stuff that makes a traditional CPU run fast. And that all those RISC MIPS don't mean much because each instruction doesn't do very much useful work. In other words, we have a difference of opinion.

***The computer industry is not based on separate enclaves of engineers working in mutual isolation, with the best bunch of engineers winning. The technology is in the air, and everybody breathes the same air.***

**A Third-Party Benchmark:**
What we need is a good third-party benchmark. We got one. Early this year, Neal Nelson & Associates ran Sun's 4 MIPS CISC machine, the 68020-based Sun-3/260, against seven RISC machines. Besides the Sun-4 SPARC, the RISC machines included the IBM RT-25 and RT-115, the MIPS M-500, the Intergraph (née Fairchild) 32C, and the HP 9000 and 825.

The benchmark—actually a series of benchmarks—tested computers' (not just CPUs') abilities to run business software in a multitasking environment. Nelson explicitly tested for 2 to 15 concurrent users. Fair enough; the seven RISC machines tested were indeed intended for a Unix or Unix-like multitasking environment.

The 68020-based Sun-3/260 is a last-generation machine, getting long in the tooth. It was the only CISC machine of the 8 machines tested. It was cheaper than most of the RISC machines tested. It was rated at only four-tenths of the MIPS performance of several of those RISC machines. *And it beat all seven of them.*

With 15 concurrent users, the 68020 beat them by a larger margin. Apparently, the people putting RISC machines in Unix boxes never stopped to think what happens when you do a context switch with a RISC machine having 200 registers. With 15 concurrent users running Unix there are a *lot* of context switches. The 68020 has to save only 16 registers—184 fewer than the SPARC, for instance.

Remember, lots and lots of registers is one of the major points in *favor* of RISC chips—and they're going to be used as the engine of Unix boxes??

These benchmarks were reported several months ago as I write this. I have been watching to see what rebuttal the five companies who make those seven RISC boxes would make. They have instead been very, very quiet, perhaps hoping no one would notice?

The RISC machines do well on small single-tasking benchmarks like the Dhrystone, where all the data will fit into the 200 or so registers. The real Unix world does not resemble the single-tasking Dhrystone.

Where do we go from here? We might look at Sun's example. Sun is a big company with (to be redundant) lots of sales. Sun got big selling 680X0 boxes, and most of its sales *continue* to be 680X0 boxes, even though it does not yet offer a modern 68030 box. When lots of folks really believed in RISC, Sun introduced the SPARC. And lots of folks seem to think that combining Unix and MS-DOS in one box is a good idea, so Sun now offers a 386 box. As soon as Sun introduces the 68030 machine that it is undoubtedly working on, Sun will have completed a CYA


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

My last article for *Programmer's Journal*, which was about fast memory and performance-enhancement techniques such as memory interleaving, was written when I was, um, unemployed.

I'm now working in the image processing industry as a hardware designer. The CRT buffer I'm designing uses an 8X memory interleave. That allows the system to transfer 16 million words per second into or out of that board while using only *half* the available bandwidth! It's a multi-ported buffer.

There are five lookup tables on the board that do pixel manipulations, and have to run at the pixel rate, which is 125 mHz. So I'm using a bunch of 1024 x 4 ECL RAMs. Their access time is 5 nsec! It's funny how life imitates art, isn't it? ♦

operation. (CYA stands for Cover Your fAnny.)

You want to buy the zillionth 680X0 box from Sun? Fine. You want a 386 machine? Fine, Sun will sell you one. You want a (choke) wonderful RISC-based Unix box? Sun's got 'em. And that's a smart move on Sun's part. If it had bet the company on just one of those product lines, it might have bet wrong.

### The Triumph of the Code Museums

The best, as in best selling, computers today are code museums. You've heard of a small computer company called IBM? What they sell are mainframes that run 360 code. The 360 was developed in the early 1960s! The most popular line of minicomputers runs VAX code. The VAX is just about the oldest 32-bit minicomputer there is. Closer to home, you have a choice of running *1-2-3* on an X86 machine or stackware on a 680X0 machine.

All of the above are code museums (a phrase originated by Gordon Bell). The manufacturers of those museums—IBM, DEC, Intel, Motorola—build spiffy new architectures every two years to hold musty old code. So every two years, that musty old code runs twice as fast.

When a new code museum—the 386, for instance—comes along, *there is an instant market for machines based on that new architecture because there's lots of musty code around!* This is also a very good thing for the owners of that musty code because it continues to sell. We do hope that the code *you* are writing *now* will be tomorrow's musty code that continues to sell, right?

Because there is an instant market for the next new code museum—the 486, for instance—those companies can afford to throw enormous sums of money into developing it. Most observers believe Intel and Motorola each spent $100 million developing their latest museum. Each version retains the good features of its predecessor and fixes some weak points. With that kind of money and that history going for them, you should not be surprised to discover that the 386 and the 68030 are very good computing devices.

You also should not be surprised when a brash young engineer or three, with *no* past history and a *lot* less than $100 million, announces the impending development of a new CPU that will be much faster than those code museums! As Countess Lovelace discovered, it's very easy to make product announcements of wonderful new computers.

Nor should you be surprised if that brash young engineer actually produces a CPU that is pretty decent. *Everybody has access to the same technology, and there is no secret to building a computer.* In fact, there are so many ASIC companies looking for customers that a drooling idiot with a seven-figure bank balance could "design" a pretty decent CPU today!

I can name seven pretty good RISC machines. All seven of them were whipped by a previous-generation CISC code museum in that recent third-party benchmark test. Are you surprised?

### Some Folks Buy Tailfins

Detroit once sold a lot of cars with tailfins. Tailfins are useless when you drive down to the corner grocery for a quart of milk. RISC machines are today's tailfins, all show and no go.

Tailfins didn't actually cripple cars. They could still be used to drive down to the corner grocery store. RISC technology, in 1988, does not actually cripple a computer. RISC


# Neural Net Models

**Introductory programs with source code *and* the compiler used to compile them**

***Netwurkz*™** A tutorial neural net model showing a simplified associative memory network. Uses DATA statements embedded in program to represent neural processing elements. "... for the price and for a flexible system, I have to recommend it." *AI SIG Newsletter*.

***Connections: The Traveling Salesman*™** A loose derivative of the classic Hopfield-Tank model with new algorithms and reduced solution time. Gives alternative approaches and bibliography. "... amazing commercially available program...Give DAIR a call." *Neurocomputers*.

***PL/D Compiler*™** A fast system language compiler and its source. PL/D self-compiles and has an extensive macro capability. It has been applied to writing neural net models, compilers, and an EPROM-based expert system. Review in July *Computer Language*.

System requirements: DOS 2.0 or above; 256K PC, AT, or compatible or PS/2. Available on 5.25 or 3.5 inch diskette. All packages include compiler object code and user manual.

To order, or for additional information, contact DAIR Computer Systems:

| Single Item Prices | |
|---|---|
| Netwurkz | $79.95 |
| Connections | 87.95 |
| PL/D Compiler | 124.95 |

| Combinations at Discount | |
|---|---|
| Netwurkz + Connections | $117.95 |
| Netwurkz + PL/D | 154.95 |
| Connections + PL/D | 162.95 |
| Netwurkz + Connections + PL/D | 192.95 |

DAIR Computer Systems
3440 Kenneth Drive
Palo Alto, CA 94303
(415) 494-7081

Add sales tax in California. Shipping: $5.00 in North America, $12.50 elsewhere. Orders shipped in 48 hours. Prices US funds on US bank or by credit card.

Number **5** on the Reader Service Card



# DEVELOPMENT TOOLS FOR GURUS, CONSULTANTS, VARS

## SAVE TIME - MAKE MONEY

Our software SHELL simplifies coding & maintenance of ***forms completion software.*** A de facto WORKSTATION SPEC defines the SHELL structure.

- Programs delivered on four disks as X/SHELL:
  - Site licensed QBOS^ OP System
    - Hosted by CP/M or MS-DOS/UniDos
    - Interprets QBIC - a Basic dialect
  - Prototype MORTGAGE FORMS PROCESSOR.
    - Illustrates the QBOS^ USER MANUAL
    - Entree to a VERTICAL MARKET
    - Customize for other applications
      - Billing, shipping, etc.
  - Plain vanilla X/BOOK wordprocessor
  - Generic OP SPEC documentation:
    - 15 Pages in X/BOOK format
    - Customize for any Workstation

Call today for more information or
Send for $25 DEMO DISK & QBOS^ MANUAL:
Your check or we ship C.O.D.
Full refund if returned in 60 days.

X/ACT
WORKSTATION SOFTWARE
JAVERT, INC.
P.O. Box 1144 Redlands, CA 92373 • (714) 793-4915

Number **202** on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

machines can still perform useful computations, if not as efficiently as a good CISC machine.

(There are *some* embedded processor applications where a single-tasking CPU with a couple of hundred registers might prove superior to a code museum with only 16 registers.)

So why did Motorola, which already has the 68030 and is working on the 68040 code museum, bring out the 88000 RISC machine? Simple. The 88000 is a CYA SPARC-killer, in case the SPARC needs killing. Did you notice that Motorola spent a lot less time, money, and resources developing the 88000 than it did the 68030 or that it will on the 68040?

**Old, Boring Subject:**

In 1982 no personal computer magazine published articles on memory caches and interleaving. In 1988 every personal computer magazine does. I don't know of anyone else who has thought about that, and considered the implications for 1994 and 2000. I do know that even the RISC supporters admit that RISC machines require twice as much disk space—40 megabytes for SPARC's compiled Unix utilities instead of 20 megabytes for the 68020, for instance. And they need larger memories and larger caches. This is for RISC's *static* memory requirements. It is the *dynamic* instruction frequency that's the killer. A RISC machine requires 34 instruction fetches to perform an integer multiply; a 386 code museum requires *one* .

RISC was born with a fatal birth defect, which is an abnormal appetite for memory bus-bandwidth. Where, one asks, does Sun propose to find 32X more bus-bandwidth than the rest of the industry in 1997?

I assume that you write X86 code for a living. What does this mean to you? Simple. You can regard announcements of fantastic RISC CPUs that are going to obsolete the Intel X86 line with amusement rather than concern. Those announcements are, literally, fantastic. ♦

---

**I Guess Speed Really Is Subjective**

I recently re-read a book I enjoyed in high school—*A Mathematician Explains* by Mayme I. Logsdon (University of Chicago Press, 1935). (No comments associating my age with the publication date, please.)

I offer the following without comment:

"There are now electrical machines which perform all of the operations of arithmetic. . . . The quotient to ten digits of a number having ten digits or less divided by a number having ten digits or less can be obtained in less than 15 sec. A square root can be obtained correct to ten places in about 2 1/2 min. . . .

"The latest models of calculators are small compact machines weighing less than 20 lb."—*W.G. Madison* ♦


# FULL AT&T C++: ANNOUNCING VERSION 1.2

Guidelines announces its port of **version 1.2** of AT&T's C++ translator. As an object-oriented language C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. "Object-oriented" means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

# C++

**from GUIDELINES for the IBM PC: $295**

**Requires** IBM PC/XT/AT or compatible with 640K and a hard disk.
**Note:** C++ is a *translator,* and requires the use of Microsoft C 3.0 or later.

**Here is what you get for $295:**

- The full AT&T v1.2 C++ translator.
- Libraries for stream I/O and complex math.
- **The C++ Programming Language,** the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Improved installation guide and documentation.
- 30-day money-back guarantee.

**To Order:**
send check or money order to:

GUIDELINES SOFTWARE, INC.
P. O. Box 749, #PJ
Orinda, CA 94563

To order with VISA or MC, phone (415) 254-9183.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

Number **204** on the Reader Service Card



# TRUE MULTITASKING

*With*

**MultiDos Plus**

"multitasking for the IBM-PC."

*Ideal* for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 programs concurrently.
- Your software continues to run under DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most languages.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication. Send/receive/check message present on 64 message queues.
  - * Task control by means of semaphores. Get/release/check semaphores.
  - * Change priority-256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!
- Independent foreground/background displays.
- Access to DOS while applications are running.

**Hardware/Software Requirements**

IBM PC/XT/AT or true clone. Enough memory to hold **MultiDos Plus** (48 KB) and all your application programs. Also may need 4 or 16 KB memory for "hidden screens" for each active task. MS-DOS (or PC-DOS) 2.0 or later operating system.

only: **$24.95** *OR* **$99.95** with source code

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order to:

NANOSOFT
13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

Number **109** on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# No-win Jobs and Too Much Work

by
**Paul A. Barkley**

**Salvage Operation**

I've been working on a system off and on for about nine months, and I've come to thoroughly dislike this particular system. I think everyone in the consulting business who also writes code runs into this every now and then. There are only a few things wrong with this particular system:

1. It's badly designed. The client told me what the system should do and signed off on the specifications I drew up. Unfortunately, everyone else in the office now tells me that the system as designed is not very useful. They have lots of better ideas, too, that they want me to use.

2. The client designed the system for a third party to use. The client commissioned this system because they viewed it as a sales tool for their own customer base; by selling it (cheaply) to their customers, they hoped to increase their own product sales. It's not clear whether a system that doesn't do what anyone wants it to do will be a viable product. It's also not clear that all the helpful new people on the client's staff with the better ideas are in fact in touch with the real needs of the ultimate users.

3. Many other systems already on the market do what this system does, and they do it much better. This system is intended for a mature market; the end users have been buying sophisticated minicomputers for years. The system I coded to the client's specs has only a fraction of this functionality.

Other than that, it's not bad. It runs well and is as clean as I can make it within the application parameters.

But I hate working on it.

I can find the most amazing little projects to undertake, rather than work on this particular system. I just wish the whole system would go away, right after they send me my last check, of course.

Two things occurred to me as I worked on this system. First, if every system were like this, I would be considerably better off as a park ranger in the Gobi desert, counting the local equivalent of gerbils. I don't need this aggravation.

Second, I realized that this situation was ripe for the kinds of lawsuits I've had occasion to observe other consultants getting into. What I would really like to do is quit working on this system, preferably right now. (This is also exactly what the important people in my personal life want me to do.) If I quit, though, the client would be stuck with a half finished, badly designed system that they might be foolish enough to ship to the end user. This would make the end user angry, which would make the client angry, and eventually the client would decide that their damn consultant led them astray and should probably pay for this mess.

This is where there's an incredible difference between being an independent consultant and being an employee. If you're an employee, you can always give notice and start looking for another job. If you're a consultant, you need to either finish the job, talk the client out of it, or find yourself a replacement the client will accept.

A couple things are important here. If we can figure out how this undesirable situation came about, perhaps all of us can avoid it in the future. More importantly, though, we need to figure out how to get some closure on matters like this. How can I salvage this particular situation?

In reviewing the chain of events, I recall telling the client that there were a number of these systems already on the market. Why write another one, particularly a short-featured one?

The client replied that they wanted a system they could virtually give away in order to get their customers to use more product and, in particular, to use the system to automatically reorder more product. (It's a variant of a point of sale system.) In addition, the system tracks some key usage statistics that assist my client in maintaining some control over the way the final product is used. (My client is a distributor and has various complicated relationships with the vendors of the product such that my client's profits are not directly tied to the volume of product, but to a tightly controlled mix of product. Whew! Neither of these two features is available from a commercial system.

O.K., that's a legitimate point. People build custom systems when stock systems can't handle their special requirements. The application makes sense. Still, most people don't like to build systems demonstrably inferior to existing systems. It certainly offends my sense of craftsmanship to build a weak product. As I mentioned earlier, the product is actually well-built—it just doesn't do as much as it could.

Well, a lot of hackers really enjoy making systems as tiny as they can. For them the goal is producing a compact, powerful product. Bells and whistles are not important. O.K., so the product doesn't sit well with me because I like my stuff to be the best, and to be full featured. I'll just have to work on that. I remind myself that I'm not in business to build stuff I like; I'm in business to build stuff my clients like.

The next point is that I made a big mistake. I assumed my contact knew what was needed; he certainly knew a lot more about the field than I did. Almost every new client takes me into an industry about which I know nothing, and I just have to trust the client's judgement about what they want. They're the experts in the field; I'm the automation expert. I can spot only the obvious goofs.

In this case, my contact in fact did not know as much as he thought he did about the way in which the product was to be used. Although he told me at the start that unlike many of the staff he worked with he did not hold a doctorate in the field, he seemed to understand the business aspects of the product, regardless of the technical aspects. As it turned out, he was wrong about some of the basic forms handling and procedures involved; in short, he really wasn't sufficiently expert in the business to be making design decisions.

This unfortunate fact didn't surface until the product reached testing stage. Although several other people were involved in design planning, management at this particular firm seems very erratic; in fact it's a rapidly growing company that has undergone several reor-



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ganizations. This wasn't clear to me when I began the project, but emerged from several design reversals in the interim. A consultant has to assume that the client can take care of their own internal problems; there's enough work taking care of their automation needs.

Thus, when I delivered the product ahead of the target schedule (but far beyond my original projections), the reviewers insisted certain features be added. In fact, the dynamics of the meeting were such that it appeared that my original contact was not precisely in charge any more and that this committee had the power to change the design. Very curious.

How can a consultant guard against this? First, ask your contact what his title is and how the project fits into the scope of the company's efforts. Find out who else will be involved with the system, and if you can word it delicately, who else might have the authority and interest to initiate or overrule aspects of the project.

In this particular case, my contact was a vice president and appeared to be completely in charge. By the time we got to the testing stage, though, he was only one of several vice presidents with authority over the project. Ouch. This is one of the risks of working with a very fast growing company. The other risk is that personnel will disappear; the primary staff resource my vice president was accustomed to turning to left long before my coding was complete.

Is the client upset that I delivered an application that won't do what needs to be done? Nope. They think it's pretty good; we just need to add this and that to it. In fact, I'm the only one who's unhappy about it.

There must be some fatal flaw in my character. I just hate to take people's money for wasted effort. The client's paid me twice for some things: once to put stuff in and once to take the same stuff out. At least they recognize that these changes aren't my fault. My contact had even balked at doing a design document before we began coding, but I refused to do anything before we designed it. Good for me.

I also protected my own backside by writing up change orders, confirming the client's oral requests to add, delete, or change features. I'm not so hardcore about paperwork that I made them sign the change orders, but they know by receiving copies of these documents (letters) that we both have a good record of the twisted application trail. If you currently don't write up change orders, do yourself a favor and start today. Word them as a casual note like the following:

Dear Bill:

Just a note to confirm your request that I remove the prompt that asks the user, "Are you sure you want to erase all your journal data for the last 5 years? (Y/N)." I appreciate your desire to keep the user's keystrokes to a minimum, and to present a clean user interface. As I pointed out in our discussion, though, this is a very bad data processing practice that has led to numerous difficulties in other systems.

I should have the safety prompt removed by next Thursday.

Sincerely,
Charlie Consultant

I might point out, though, that if you're writing code that controls a nuclear reactor, writing such a note will not absolve you from legal difficulties in the event the reactor melts down and spreads radioactivity over half of Europe. Lawyers call that sort of stuff "Strict Liability"—one reason a colleague of mine, wiser and more paranoid than I, refused to help me write a system to control the ultraviolet lights above the beds of a local tanning salon. He figured that sooner or later there would be a power surge and our code would fry someone well done instead of just medium rare.

Anyway, use change orders primarily to leave a paper trail of the changes the client requests after the task begins.

I think one of my biggest mistakes, though, was in not recognizing that this system was being designed by one party for the use of another party. That's always risky. It's hard enough for the client to figure out what they want, without trying to figure out what a third party wants. That's what they pay MBAs for, to figure out what people want (and think they need).

O.K., so how am I going to get some closure on this? We're looking at a shipping deadline of about 5 weeks. That should do it. After the latest round of changes are incorporated, I'll announce that we're out of time to make any further changes. The way it is, is the way it will be shipped. Of course, I don't have the line authority to make that decision, and I'll be a little more tactful about it, but before anyone looks at the last round of changes, I'll make the announcement. Of course, I'll clean up any bugs in the code, but otherwise there won't be time for any more changes.

The next thing I'll do is close out my case and put these folks down to the end of the line. That means, in this case, a delay of approximately three months before I can get to them again. I suspect that if they want more changes, they'll try to gather them all together instead of dribbling them in to me one or two at a time. I am very fortunate (actually it feels like cursed, more about this later) in having a considerable backlog, so I can get away with this tactic. When you're scrambling to make ends meet, you want to accelerate client assignments, not delay them.

So what's the moral of this story? This is a system I grew to loathe, for various emotional reasons, but didn't abandon because of a professional commitment to do the job. I got paid regularly, and although I'll never like the system, I stayed out of court and delivered a workable product. End of moral. Bah.

What kinds of systems do *you* despise? What gets your dander up? What sort of work do you just hate to do, and how do you manage to stay away from it?

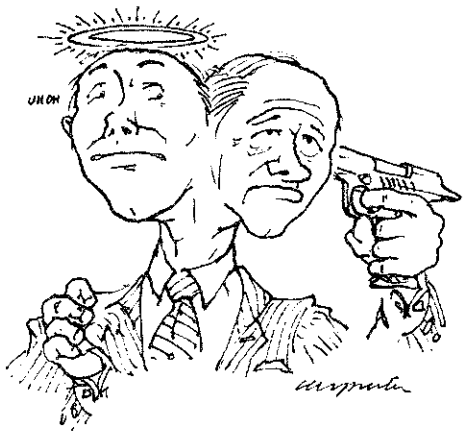
**Backlog**

I mentioned earlier that my backlog is now about three months. I imagine it won't grow much longer than that because most clients can't wait forever to get systems worked on. Plenty of other consultants out there could pick up the slack.

If you've been consulting for awhile, perhaps you have the same problem. If you're just starting out, you're probably finding it real hard to be sympathetic with this "problem"; what you need is a lot more business.

Actually, for years I was never able to create much more than a two week backlog because I never had any very large projects to work on. Everything I did was small, usually lasting only a month at most, and often overlapping with other client work. I don't work on site (one of the advantages of being a micro rather than a mainframe consultant), so by judiciously mixing tasks I was able to have several clients active at one time. A lot of time is wasted waiting for clients to respond to queries and make decisions, so it's seldom possible in this type of consulting to get 40 hours a week from any one client except during brief spells of pure coding effort.

I've said repeatedly, though, that the most successful (that is, the most gainfully employed) consultants I know are the ones with the largest contracts. It took me years to figure this out. (Hey, sometimes I'm a little slow; I'm sure you knew this all along.) I'm not talking about contract programming or similar work, where the "consultant" shows up on site as do all the rest of the employees. I'm talking about contracts at





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

**DOS & OS/2**
*Free Demo Diskette Available*

**Q. How many programmers does it take to maintain a MAKE dependency file?**

**A.** NONE! *If you use* VersiMAKE™

VersiMAKE™ is a full-featured MAKE utility that includes:

- **Dependency Generation**
  Derives your system's dependencies, through analysis of its C and MASM source files. Say goodbye to manual maintenance of MAKE dependency files!
- **Wild Card File Name Matching**
  Analyzes an entire collection of source files with a single statement.
- **Nested Include Files**
  Handles standard C and MASM "include" conventions, and the INCLUDE environment variable.
- **Powerful Macro Facilities**
  Built-in macros, user-defined macros, and environment variables.
- **Analytical Reports**
  Shows the entire Include file hierarchy for each source file analyzed, and all of the parent source files for each Include file.

**Q. How many programmers does it take to trace a symbol thru your system?**

**A.** ONE! *If you use* VersiCREF™

VersiCREF™ is a unique utility that creates a Master Cross-Reference of your entire system.

- **Multi-Lingual**
  Handles C, assembler, or both.
- **Flexible**
  File names with line numbers, or file names alone. Global and local symbols, or globals alone.
- **Powerful**
  Easily handles systems containing 100 source files or more.

| | |
|---|---|
| VersiMAKE™ | $125 |
| VersiCREF™ | $75 |
| Both | $150 |

*(Outside U.S., add $5 for shipping and handling)*

**800-334-4096**
(In NJ, 609-871-0202)
MC/VISA/AMEX accepted

SUMMIT INFORMATION SYSTEMS, INC.
73 East Lane, Willingboro, NJ 08046

Number 158 on the Reader Service Card

$10,000 to $20,000 (or more) instead of $2,000 to $5,000 at a crack.

The size of contracts you get seems very closely related to the sorts of clients you cultivate. One successful consultant I know specializes in small government procurements; his clients are all agencies looking for chunks of contract help at $10,000 to $25,000 a pop. He does some on-site work for them, particularly in LAN configuration and troubleshooting, but a lot of his work is applications development. Another successful consultant I know stays away from government contracts, but has larger corporate clients. I haven't figured out how he manages to make so many new contacts, but one of the industries he specializes in is banking, and the banking community seems to be pretty clubby. (Banks themselves are not the clients, though.) Because his clients know a lot of other potential clients in cities outside their competitive areas, he often gets referrals. By contrast, clients in some of the industries in which I've consulted harbor incredibly hostile attitudes toward their fellow practitioners. That probably has to do with the fact that everyone they know is either a direct or indirect competitor; this might radically affect my attitude, too.

In my case, I mentioned that I finally took a full time position on a limited duration project, which of course has continued to grow. Due to the Section 1706 brouhaha, I'm officially an employee, and it certainly feels that way. My title says I'm a consultant, and there isn't too much job security there, but there's no getting around the fact that I act as an employee would and am treated that way. Someone I work with insists on telling people he's always in a consulting "mode"; that's a polite way of saying that he wants to feel like an independent consultant and somehow feels ashamed of being an employee.

Anyway, I've found that this chunk of full-time work has completely upset the apple cart. I've had slow spells in the consulting business before, and several times in the last six or seven years I've gone an entire month without any billable hours (gulp!). As I prepared to take this job, I made an effort to notify my active clients I'd be tied up for an extended period, and that we should finish up any work they wanted to get done. I was swamped. I worked like a maniac to get everything done, and finished the last piece after 11 p.m. the night before I started my new job. What a feeling of accomplishment.

I was pleasantly surprised in that while I was learning the ropes the first few months, I had no further "private" business. It was absolutely dead. I congratulated myself on having the foresight to anticipate the drought and take the full time position even though the two months prior to my joining the mainstream work force had been the best two months of my entire consulting career. That seemed like a strange time to curtail my consulting, but then in card games they say you should quit while you're a winner.

I immediately noticed how easy it is to be an employee. I could do a regular 40 hour work week standing on my head; it's incredibly easy compared to consulting. Every two weeks when they come around and hand me a paycheck, I have to shake my head in disbelief. Incredible. I don't even have to chase my money.

**The Inevitable**

But then it started happening. I got a few calls from old clients. Could I add a report here, and change a calculation there? Sure. And now the money was just gravy. Everything that came in from my private work was completely disposable; the regular job took care of everything else.

Then I made a big mistake. I got a call from a government agency who was *not* a previous client, and they asked me to attend a meeting. O.K., I combined it with a business trip and worked a few hours late that night to make up for the time taken. The agency wanted me to bid on a feasibility study. The other bidders were representatives from a couple large companies who are competitors to my present employer. Holy Conflict, Batman!

So I took the paperwork to my bosses and told them I had a chunk of business I thought I could bring into the company and asked for their O.K. to proceed on this. I outlined the strategy I thought would win the business. I made sure I mentioned the fact that our competitors were bidding, and that I recognized the conflict of interest if I were to bid on it myself when they might consider it.

They had to have a big powwow and decide whether they wanted this business or not. It concerned a government agency who was not one of their current clients, and so it seemed like an ideal foot in the door to me. I was pleased to be able to offer this to my employer and figured that if I stayed on after the current project was completed that this wouldn't hurt my status.

No dice. The decision was made that we didn't have anyone we could spare to do the work; I was judged indispensable on my current assignment. Also, they said the work was too small and didn't fit the company profile. In fact, that's what the competitors were complaining about at the bidders' conference. How small was this contract? No more than $25,000.

Now, when you work in a company that routinely gets multi-million dollar contracts, anything less than $25,000 does seem pretty small. But if you're my size, this looks like a lot of money to be throwing away. It depends on your perspective.

I wound up bidding the job with another consultant who was willing to do all the on-site work such as interviews; I was not free to wander around during the day. I was willing to do all the analysis and report writing if my partner could just gather the data. I was not surprised when we won.

The upshot is that it was a large chunk of business, especially on a part time basis. It consumed most of my free time for several


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

months. I made a lot of money; it was bid at my top rate and was all "disposable" income. I was happy with the financial results, and the client was happy with the report, although along the way my partner and I had some unhappy moments.

We didn't meet with each other as often as we said we were going to, and my partner had to go back many times to gather additional information I needed. It was the first time he and I had worked this way; the "preconsulting" data gathering was a little out of his field although he's a good interviewer, and more patient than I. Consequently, our study was not as efficient as it could have been. We also found that we had walked into a political situation we hadn't suspected when we bid the job. A very high level of management had insisted that the study be done for a department that deeply resented our intrusion. They didn't much like our recommendations, either, but upper management was very satisfied with the results.

So what does this have to do with consulting in general? What do you suppose happened when I started this large project (part time, yet)? Yep, other work started trickling in, and I got behind. Real behind.

Old clients started calling up with a few things here and there. Then one client experienced a catastrophic spurt in business and needed to have his old system completely revamped. Within the space of a few months the backlog grew, due mostly to the one large project. With the supply of product (me) reduced, the regular demand was sharply increased by just a single project, causing a massive jam-up, something like rush hour on the Washington beltway.

But isn't this just what the average consultant wants? Sure! The problem most consultants have is too little business, with some of that business not very profitable to boot. The goal is to get a nice steady flow of profitable business. The reality is that you can never keep the flow steady; you need to get the flow slightly backed up so that by the time you get caught up something new will have arrived.

So what am I complaining about? Well, I said that you need to get it "slightly" backed up, not all screwed up. Unfortunately, I've reached a point where I'm having trouble properly estimating when I can even begin a job because the smallest addition to an ongoing job throws everything off; there's no "slack time" to make up for overages. If a client wants to add something, it's difficult to tell them they'll have to wait three months—when you're already in the middle of their system. If you keep adding stuff, though (we're assuming either an hourly basis or change order cost increases on a fixed fee basis), you might never get to the next job. It's a tough problem.

The other thing I didn't realize is that this is just plain stressful. I don't mind working a lot; most consultants I know are pretty flexible in this regard because they know they need to take care of business when it's avail-


FINALLY!

# EASYFLOW

*An on-screen flowchart processor that knows about flowcharts - not just another "screen draw" program that makes you do most of the work. EASYFLOW is a powerful full-screen graphics program dedicated to flowcharts and organization charts. With it you can quickly compose charts. More important, you can easily modify charts so they are always up to date.*

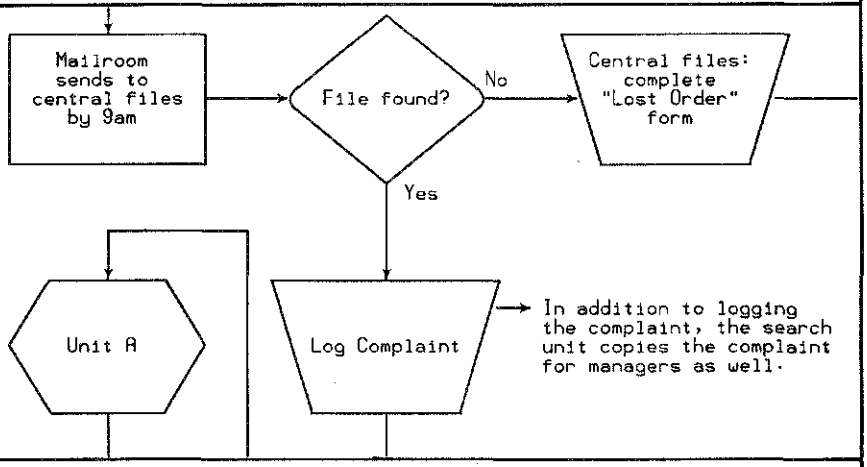
- **Automatic:** Fully automatic text centering within shapes, both horizontally and vertically. Fully automatic line routing & re-routing.
- **Fast:** Written in assembly language for speed.
- **Large:** Charts up to 417 columns wide by 225 lines high. Chart too large for your printer? *EASYFLOW* automatically breaks the chart up & prints it in page size pieces.
- **Standard:** All standard flowcharting shapes included. Custom shapes can be ordered.
- **User friendly:** Don't take our word for it. PC Magazine* says "*EASYFLOW* lives up to its name. It's hard to imagine any easier and more flexible way to produce basic and even complex flowcharts".
- **Mouse:** Optional but fully supported.
- **It prints:** On most popular matrix printers including IBM, Epson, Toshiba, HP LaserJet, LaserJet-Plus and many others.
- **It plots:** On HP7440, 7475, 7550, 7570, 7585B and compatible plotters.
- **It works:** We are contractually prevented from mentioning the name of the "big eight" accounting firm that purchased a world-wide site license, but we can tell you that they spent months evaluating all available flowcharting packages before choosing *EASYFLOW*.
- **Rush delivery:** Order by noon today (eastern time) and we'll have it to you by courier tomorrow**. Rush delivery charge is $15.00 (instead of $2.00) and is available only in USA & Canada.
- **Documented:** 100 page manual plus over 150 screens of context sensitive help.

*EASYFLOW* works on IBM PC's, IBM PS/2 and compatibles. Requires 384 K memory, DOS 2.0 or higher and an IBM CGA/EGA/VGA or Hercules monochrome compatible adapter card.

Order direct for only **$149.95** + $2.00 S&H (USA/Canada), $10.00 (foreign). Payment by M.O., cheque, VISA, Mastercard or Company PO.

* March 10, 1987 issue, page 278.

** Rush orders are shipped by Purolator Courier and normally arrive the next business day to most locations. Remote destinations take longer.


Mailroom sends to central files by 9am
File found?
No
Central files: complete "Lost Order" form
Yes
Unit A
Log Complaint
In addition to logging the complaint, the search unit copies the complaint for managers as well.

The chart fragment above was produced on an HP LaserJet-Plus and is actual size and unretouched. Publication quality charts like this can be produced using only minutes of preparation time and seconds of print time.

**HavenTree Software Limited**
PO Box 1093-V
Thousand Island Park, NY 13692

Order Desk: 1-800-267-0668
Information: (613) 544-6035 ext 31
Telefax(G3): (613) 544-9632

Number **154** on the Reader Service Card


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

able; you never know how much more there will be. But everyone I know wants at least some free time, say one or two nights a week and maybe a Saturday or Sunday off.

This is a problem you need to think about in your own practice. What happens when you get too much work? If you make rash promises you can't keep, you won't have your clients very long; and if these promises compromise the integrity of your product, you may wind up in court, too. When this happens, your productivity will go downhill and things will get even worse.

You may also discover that you've become quite unpopular in your family. There are a lot of ways to break up relationships, but this is a pretty good one. Can your relationship(s) handle this?

The obvious solution is to proact instead of react. Plan how you're going to deal with this situation.

One method is to off-load work to your associates. There are two problems with this. First, you lose the revenue. Some consultants have informal networks where they regularly off-load work they don't have the time to handle; they hope that in some future slack time the person they gave the work to can reciprocate and send some work back. Most consultants I know are not greedy, but they've struggled pretty hard to learn their trade, and they're loathe to give up any work. I've taken work to other consultants in the past for a finder's fee, usually 5%. It was worth it to me to have the other consultant remember me in the future—even though many times the other consultant went out of business before the next pass-on work arrived. If you're giving away work, though, 5% isn't much money. You'd have to be virtually a broker to get anywhere, and that takes volume. I doubt that anyone has that much work to give away.

Another method is to train an employee or trusted subcontractor. A subcontractor is tough to work with because he or she is usually a consultant, too. That means the subcontractor probably wants as much money as you're charging already.

So how about an employee? What a can of worms! First off, there's a lot of paperwork and hidden costs such as unemployment insurance to worry about. It's not prohibitive, but it does take time. Next, where will the employee work? In your basement? At his or her own house? Do you need to buy some equipment? (Probably.) How are you planning to train the employee if you're so busy already that you can't do the work?

***I'm still not ready to give away work; I've grown quite fond of the cash flow from all of this.***

I'm still not ready to give away work; I've grown quite fond of the cash flow from all of this. On the other hand, not one of my projects has been large enough to warrant the time needed to train an employee. I have no place to put an employee because I closed my regular office when I took the full time position. Worse, I'm not sure I have enough work coming in to keep an employee busy. When I was a full time independent, I spent a lot of time hustling up business. I can't do that now, and I can't expect an employee to do that.

If your practice is growing, though, you might want to take on an employee as soon as possible (meaning, as soon as you think you've got enough work coming in). If you're a sole practitioner, you have a real problem. The consultants I know who have hired employees did so on the premise of getting a large contract (say, $25,000) in addition to their regular on-going work. They took a chance and hired the employee before they really needed him or her. This worked out for the people I know; the people that it didn't work out for probably went out of business before I met them.

The problem for the sole practitioner is that the first employee represents a 100% increase in manpower. No additional employee will ever affect the firm's staffing as much as the first. (The second causes a 50% increase, the third a 33% increase, while the nineteenth employee causes only a 5% increase.) That's truly an awesome leap. That first employee is the one who causes you to have to start filing forms and withholding taxes; after the first it's just more of the same.

I discussed this phenomenon with a fellow consultant during the first few months after he hired his first employee. He was lucky and won the contract he hired the employee for. (The big time government contractors, of course, always wait until the contract award to start hiring. A company like this can grow from 50 employees to 500 employees in a month or two; you can imagine how interesting that is.)

Although he was able to find a fairly competent employee, the consultant told me repeatedly how frustrated he was that the employee wasn't as good as he was. He constantly had to correct programming bugs and watch out for style problems. He let a second employee go soon after hiring him, perhaps because he was hired too soon, but mostly because he "didn't work out."

A big problem with hiring anyone at all is cash flow. That first employee had to be paid, but until he had done several weeks' worth of work, the consultant could not bill the client. It then took the client a good 30 days to pay the bill. The result is that the consultant had to pay the employee for 6–8 weeks before the first money started coming in. If you hire a programmer for, say $30,000, that means you'll need to have socked away something like $6,000 for salary and your


Form-Fill Toolkit

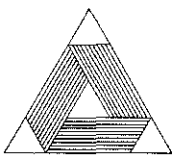
Product: C Library
Features:

Price: $239.95
Includes Source Code

Cursor Control
Sound & Printer Control
Fast Screen Access
Keyboard Access
Edited Field Entry
Pop-Up Window Support
Display Text Windows
Menu & List Selections
Form-fill Routines

Compiler Support
Microsoft 4.0, : Y
Microsoft 5.0 : Y
Quick C : Y
Turbo C : Y
Other:

↑↓ Scroll F1 Help F2 Save F4 Cancel Esc No Save



Provide a consistent user interface for your programs with over 100 documented functions.
***SPECIAL! Includes powerful multi-record data collection module with fully commented pristine source code.***

Anji Software, Inc.
P.O. Box 297
Cedar Grove, NJ 07009
(201) 239-4328

- **Ask for our Demo Diskette**
- Specify compiler when ordering
- No royalties required
- **30 Day Money Back Guarantee**

We accept Visa and M/C
NY & NJ residents add sales tax
Shipping: $3.00 Domestic
$10.00 Foreign

Number **162** on the Reader Service Card





PRODUCED 2006 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

share of taxes and benefits, probably more. That's the bad news.

The good news is that this consultant bills the programmer out at $55 to $60 an hour. Hmmm. It doesn't take a rocket scientist, as they say around here, to figure out that you can make some decent money that way. Your cost per hour works out to something like $20, so you make $35 to $40 an hour on someone else's time. Say, that's as much as $1,600 a week profit. After the initial 8 weeks of cash flow drain, it only takes another 4 weeks at this rate to break even and start making money.

Do you suspect there's a catch here somewhere? Yep. First off, a lot of contracts, especially government contracts, won't let you bill out a $20 an hour person at $60. In many parts of the country, you might be able to hire a decent (junior) programmer for less than this, but you can't bill him or her out at anywhere near $60. $40 an hour might be more realistic.

The other problem is that you probably can't bill the employee out at 40 hours a week unless it's a real long term contract. There will probably be a lot of incidental work that you can't (or shouldn't) stick on someone's bill. Couple this with employee vacations where you pay the salary but don't get any work, and you can see that your cash flow might get a little more erratic.

In time, though, this consultant built his business up nicely. He was able to generate so much profit per hour on the employee that the time spent supervising him seemed worth it. More importantly, after about 6 months of this, the business had stabilized enough that the consultant could resume paying himself a salary. (I forgot to mention that the cash flow was bad enough for a while that his own salary had to be foregone; savings and the spouse's earnings carried the day. You *do* have savings and a spouse, don't you?) Now this consultant pays himself about $70,000 a year, certainly not outrageous in an area where the average new home price is $174,000.

Think about your own practice and how it's (I hope) growing. Are you comfortable with the idea of an employee? Can you afford one? Are you willing to take what may be a dumb chance to acquire one in the hopes of more business? Think about it before it gets to be a problem. ♦

*Send your comments or questions to Paul A. Barkley, 6845 Elm Street, Penthouse One, McLean, VA 22101.*

---

**Did You Know . . .**

The first true programmable "digital" computer, Babbage's Analytical Engine, cost the British government over £100,000 (a lot of money in those days) and was never completed?—*W.G. Madison* ♦


WITH 'C' SOURCE

For UNIX* and DOS

*C-ISAM is a trademark of Informix Software, Unix a trademark of AT&T.

'W'

MAKES CURSES obsolete!

- 'W' is a windowing library for 'C'
- 'W' gives you prioritized windows
- 'W' offers complete control of any terminal
- 'W' drives the DOS Memory map
- 'W' has line graphics and bells
- 'W' has soft keys
- 'W' offers 'hot key' control
- 'W' prints screens even in UNIX*
- Describing terminals is much easier with 'W' than termcap
- It's easy to port from curses to 'W'
- $295 (U.S.), you get SOURCE for UNIX* & DOS

D-ISAM

C-ISAM* Compatible

UPCOMPATIBLE WITH C-ISAM* VERSION 2.0

- Both use compatible function calls
- Both use compatible datafiles
- Both work the same files at run-time

WE ADDED:

- Extended types and natural ints
- Strings
- Improved file integrity checker

D-ISAM:

- A complete (B+tree) filing system
- Manages multiple keys
- Uses read only record locking

PRICE INCLUDING SOURCE:

- $595 (U.S.) for UNIX* & DOS

BYTE DESIGNS Ltd.
P.O. Box F195-76
BLAINE, WA 98230
# 6-3511 Viking Way, RICHMOND, B.C.
V6V 1W1 CANADA
Toll-free 1-800-663-8547
PH: (604) 278-5200 FAX: (604) 278-4045

Number 219 on the Reader Service Card

NOW AVAILABLE IN C TOO!

Write A Data Base Program
*(end to end)*
in 10 minutes in TPascal 4.0
using Turbo Programmer
(formerly Turbo GhostWriter)

Perfect for creating complex business applications!
Ideal for BASIC programmers who find TPascal too tough!
Spec your customer in the morning - Show a demo in the afternoon!

**Features**

- screen editor and painter
- automatic programmer documentation
- automatic data checking/validation
- plenty of "hooks" for customizing
- unlimited technical support

**More features**

- automatic B-tree indexing
- consistent user interface
- automatic context-sensitive help
- relational model to show customizing
- 30 day money-back guarantee (less $14 shipping/handling)

No questions to answer. Just draw your screens the way you want them to appear, tell Turbo Programmer how to set up the indexes and that's it... Running 4.0 code in 6 seconds with no programming. Regular price $450.

Now Only $289 Orders & Info 800 227-7681

ASCII 3239 Mill Run, Raleigh, NC 27612-4135

Number 170 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# VGA Text Paging

by
**Michael Abrash**

*Installment 8: In which we explore the text-paging capabilities of the VGA and find 8 extra pages by moving display memory and 16 more pages hiding in the Page bit, for an amazing total of 32 text pages.*

Now that we've covered all the read and write modes of the VGA, we'll have to cast about for a new graphics topic. We won't have to cast too far, though, given the power and complexity of the VGA; the real problem is selecting among the many possibilities. Andrew Chalk, of Magna Carta Software, suggested that I discuss the VGA's text paging capabilities; that sounds like a good idea to me, so that's what we'll look at in this installment.

### A Quick Overview of Text Paging

Text paging is the technique of storing information for two or more screens of text simultaneously in separate areas of display memory, then flipping between those screens virtually instantaneously by changing the portion of display memory that the CRT controller (CRTC) circuitry in the display adapter is sending to the screen. Figure 1 illustrates the basic operation of page flipping. In Figure 1a, the CRTC is programmed to fetch video data from the page of memory starting at offset 0 in display memory, so page 0 is displayed. In Figure 1b, the CRTC is programmed to fetch video data from the page of memory starting at offset 1000h, so page 1 is displayed. The key point is that two text screens can be displayed in turn *without the contents of display memory being altered in any way.*

Two basic requirements must be met in order for a given graphics adapter to support page flipping. First, the adapter must have enough memory to store information for at least two screens at once. Second, the adapter must be programmable as regards the address at which it starts scanning display memory for video data.

The CGA, EGA, and VGA all meet the above requirements and support video paging while the MDA, which has only 4K bytes of display memory (just enough for one screen), does not. Since the Hercules Graphics Card has a programmable start address and has 64K bytes of display memory (and can, in fact, perform page flipping in a different way in graphics mode), it's theoretically capable of text paging, but my experiments indicate that the HGC does in fact provide only the single text page that the MDA supports. This makes some sense, in that it provides maximum MDA compatibility.

The CGA has 16K bytes of display memory; since one text-mode screen requires 4K bytes of display memory, the CGA supports 4 80 x 25 text pages. The CGA also supports 8 40 x 25 text pages since a 40 x 25 text screen requires only 2K bytes. The EGA and VGA similarly support twice as many 40 x 25 as 80 x 25 text pages, but since 40 x 25 text is rarely used anymore, I'll ignore it for the remainder of this article.

CGA text paging is straightforward; the 4 pages are laid out one after the other in display memory as shown in Figure 2, and a given text page is selected for display simply by programming the Start Address High and Start Address Low registers of the 6845 CRTC (indexed registers 0Ch and 0Dh at I/O port 3D5h, respectively) to one-half the offset of the start of the desired text page in display memory. (The one-half is the result of the CRTC's characteristic of counting in units of character/attribute pairs in text mode rather than in units of bytes of display memory.) The code in Listing 1 flips through the 4 text pages of the CGA, advancing one page each time a key is pressed.

Each CGA text page is actually 4000 rather than 4K bytes long (80 columns times 25 lines times 2 bytes—character code and attribute—per character); 96 bytes are wasted at the end of each page. This means that you could, if you wished, pack the pages more tightly by starting them at addresses 0, 4000,

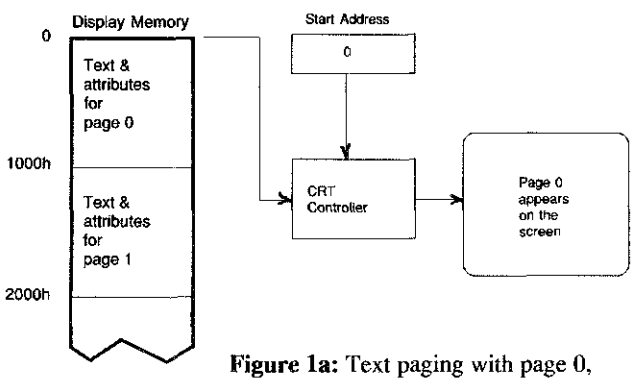



**Figure 1a:** Text paging with page 0, starting at offset 0 in display memory, selected.

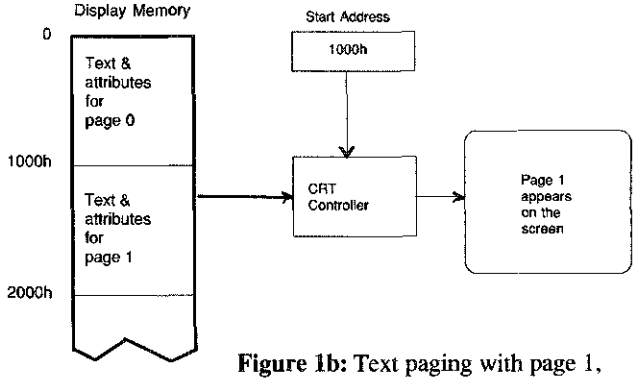
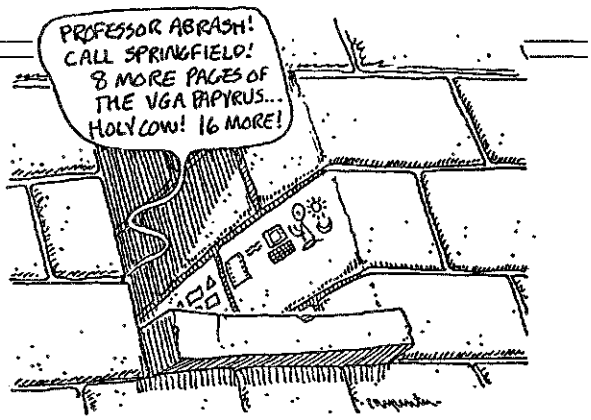



**Figure 1b:** Text paging with page 1, starting at offset 1000h in display memory, selected.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




8000, and 12,000 instead of the addresses 0, 1000h, 2000h, and 3000h that we used above. It doesn't much matter because either way there's room for only 4 text pages; since IBM's BIOS uses 1000h for the text page length, that's what I'll use as well.

By the way, you may get the clever idea of using those extra bytes of display memory to store program data (or even code, if you're really clever); why let good memory go to waste? Don't do it! Display memory is much slower than normal system memory due to the wait states inserted while the CRTC fetches video data. In fact, an AT can slow down to PCjr speeds when accessing display memory, so don't use display memory for anything other than video data unless you absolutely have no alternative.

That's really all there is to CGA text paging. One handy feature of CGA text paging is that the BIOS supports text paging to some extent, maintaining a separate cursor for each text page and allowing you to write text to and scroll whichever page you've selected to be active. Both the EGA BIOS and the VGA BIOS support the text-paging BIOS functions as well. (In fact, the EGA and VGA BIOSes support all CGA BIOS functions.) Beware, though—I've heard that some versions of the IBM BIOS have bugs in their active page handling code, so test your code on several machines before you rely on the BIOS for text paging. You might also want to take a close look at the video BIOS listings in IBM's PC, XT, and AT *Technical Reference* manuals if you're using BIOS text paging, to make sure the BIOS code really does what you expect it to. I'm not going to discuss the BIOS functions related to text paging here since they're documented in several of IBM's manuals, including the aforementioned *Technical Reference* manuals and the *BIOS Technical Reference* manual.

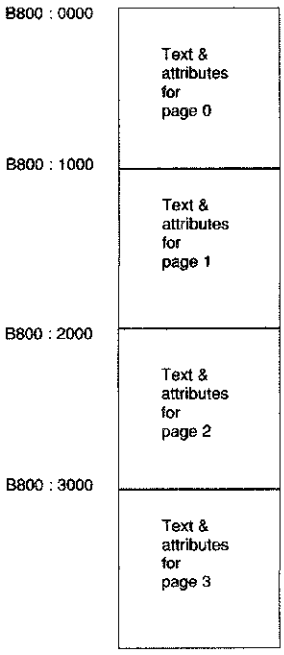

B800 : 0000
Text & attributes for page 0
B800 : 1000
Text & attributes for page 1
B800 : 2000
Text & attributes for page 2
B800 : 3000
Text & attributes for page 3

**Figure 2:** The 4 text pages of the CGA.

Be aware that the cursor location is programmed with an address, not a screen position; that is, the cursor isn't positioned by setting an (X,Y) coordinate. Instead, the Cursor Location High and Cursor Location Low registers (CRTC registers 0Eh and 0Fh) are set to cause the cursor to appear when a specific memory address is read by the CRTC for video data. The cursor can be made to appear in any page by setting the cursor location to fall within that page. Cursor locations correspond directly to start addresses; for example, a cursor location setting of 800h would position the cursor at the first character of page 1, just as a start address setting of 800h would cause page 1 to be displayed. Likewise, a cursor location setting of 1001h would position the cursor at the second character of page 2. (As with start addresses, a given cursor location value is one-half the value of the CPU address of the corresponding page. Page 1 starts at offset 1000h, but the cursor location required to position the cursor at the start of page 1 is 800h.)

The EGA and VGA are by far the most interesting adapters as far as text paging is concerned since each is capable of supporting up to 32 80 x 25 text pages. Now, given past experience, you might well guess that text paging is more complex on the EGA and VGA than on the CGA—and you'd be quite correct. The first 4 pages are the same as the CGA's, and the next 4 pages lie just above the first 4 pages, but the remaining 24 pages are somewhat harder to find. It's just a matter of knowing where to look, though, and that's what this installment's about . . .

By the way, while the VGA does have text-paging capabilities that the EGA lacks, everything discussed in this installment will work on both the EGA and VGA, so from now on my references to the VGA imply the EGA as well, unless otherwise noted.

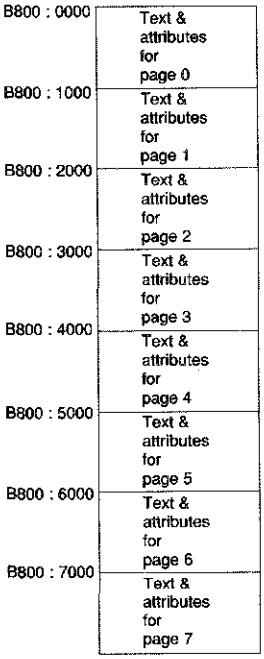

B800 : 0000
Text & attributes for page 0
B800 : 1000
Text & attributes for page 1
B800 : 2000
Text & attributes for page 2
B800 : 3000
Text & attributes for page 3
B800 : 4000
Text & attributes for page 4
B800 : 5000
Text & attributes for page 5
B800 : 6000
Text & attributes for page 6
B800 : 7000
Text & attributes for page 7

**Figure 3:** The 8 standard text pages of the VGA.

### The 8 Basic Text Pages

In text mode, the VGA offers 32K bytes of display memory starting at B800:0000 rather than the 16K bytes of the CGA. Consequently, the VGA readily supports 8 text pages in just the same way that the CGA supports 4, as shown in Figure 3. Accessing one of the 4 pages that the VGA adds (pages 4–7) is simply a matter of reading from or writing to the appropriate address in the range B800:4000 through B800:7FFF in display memory, and displaying one of the 4 pages merely requires the use of the appropriate start address for that page (2000h, 2800h, 3000h, or 3800h).

In order to see the 8 basic text pages of the VGA in action, just change the value of NUMBER_OF_PAGES in Listing 1 from 4 to 8. Incidentally, Listing 1 illustrates an important concept in text paging: At any one time, the page that your program is currently modifying does not have to be the page that is being displayed. In fact, the opposite situation is often


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

desirable since you generally won't want the user to see a given page while you're in the middle of updating it. The cleanest approach is to display one page, redraw a second page, and then switch to displaying the second page once it's fully updated.

**Moving Display Memory Turns Up 8 More Text Pages**

We're limited to 8 text pages in the range B800:0000 to B800:7FFF because the VGA supports only 32K bytes of CPU-addressable display memory when display memory starts at B800:0000 (as is the case in all BIOS-supported color text modes). Now, it happens that the VGA supports 64K bytes of CPU-addressable display memory in those modes—0Dh, 0Eh, 0Fh, 10h, and 12h—in which display memory starts at A000:0000; unfortunately, those modes are all graphics modes. What we'd really like to do is reconfigure display memory in text mode from 32K bytes at B800:0000 to 64K bytes at A000:0000, which would give us a total of 16 text pages.

Does the VGA let us do that? Sure. Bits 3 and 2 of the Miscellaneous register (Graphics Controller register 6) select the location and length of the VGA's display memory as follows:

| Bit 3 | Bit 2 | Display Memory |
|---|---|---|
| 0 | 0 | Starts at A000:0000; 128K bytes long |
| 0 | 1 | Starts at A000:0000; 64K bytes long |
| 1 | 0 | Starts at B000:0000; 32K bytes long |
| 1 | 1 | Starts at B800:0000; 32K bytes long |

Normally, bits 3 and 2 are both set to 1 in color text modes. All you need to do in color text mode in order to make display memory 64K bytes long at A000:0000 is change bit 3 from 1 to 0. Display memory will instantly become addressable in the range A000:0000 to A000:FFFF, organized in just the same way as it was in the default range B800:0000 to B800:7FFF, and 16 text pages will become available.

The text pages are located just where you'd expect, as shown in Figure 4. Likewise, the start address settings for displaying the text pages are a straightforward extension of the start addresses we've already encountered. Basically, text paging at A000:0000 is just like text paging at B800:0000 except that the segment address is different and display memory covers a greater memory range. In order to see the 16 text pages of the VGA in action, modify the value of NUMBER_OF_PAGES in Listing 1 from 8 to 16 and modify the value of RELOCATE_TO_A000 from 0 to 1.

Setting RELOCATE_TO_A000 to a non-zero value has two effects on the program in Listing 1. First, a non-zero setting for RELOCATE_TO_A000 causes MASM to assemble a section of the program that instructs the VGA to move display memory from the segment at B800 to the segment at A000 by way of the Miscellaneous register. Second, a non-zero setting causes the program to set ES to point to the relocated display memory at A000.

By the way, the 128K byte display memory configuration noted in the table is an interesting aspect of the VGA. While the VGA is indeed capable of supporting 128K bytes of CPU-addressable display memory, no BIOS mode (and virtually no software that I know of) supports this memory configuration, and there are some good reasons for that. The first reason is that there's really nothing you can do in the 128K mode that you can't do in the 64K mode. (You may be thinking that we could get 32 text pages if we could address 128K bytes of display memory, and that's true; as we'll see shortly, though, we can also get 32 text pages with just 64K bytes of display memory.)

The second reason the 128K mode isn't widely used is that the 128K mode flat-out can't work in any computer that has two display adapters because a CGA, HGC, or MDA would try to respond at the same memory addresses as portions of the VGA's 128K bytes of display memory and neither adapter's memory would work reliably in the overlapping address range. (Of course, PS/2's don't support CGAs, HGCs, or MDAs, so the 128K mode could well be employed there if we had a good reason to use it.)

At any rate, the 128K mode exists, and perhaps someday I'll get around to discussing it in detail and writing some sample code, but on the whole there are complications to the use of 128K mode, and relatively little to be gained by using it.

So far I've discussed only color text paging. On the VGA, monochrome text paging (that is, text paging in mode 7) works in much the same way as color text paging, save only that the 32K bytes of monochrome text mode display memory normally start at B000:0000 rather than B800:0000. Monochrome text mode display memory can be moved to A000:0000 just as color text mode display memory can—by setting bits 3 and 2 of the Miscellaneous register to 0 and 1, respectively.

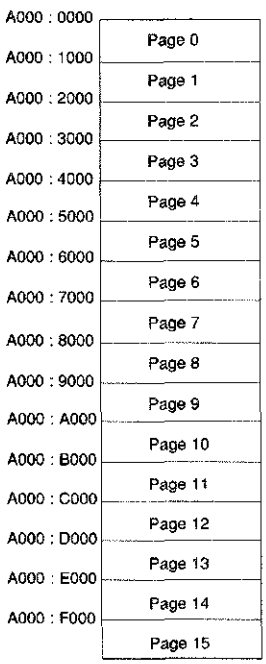



**Figure 4:** The 16 VGA text pages that are available when display memory is moved to the segment at A000.

One unfortunate result of moving text mode display memory to A000:0000 is that the BIOS no longer provides support for placing characters on the screen; the BIOS assumes that monochrome text mode display memory starts at B000:0000 and that color text mode display memory starts at B800:0000 and can't be convinced otherwise. The solution is to follow the approach I've used in Listing 1, accessing display memory directly to place characters on the screen rather than going through the BIOS **write_tty** or **write_character** functions. While this is a bit more work, you'd probably want to modify display memory directly in any case since code that accesses display memory directly can update the screen much more rapidly than can code that uses the BIOS functions. After all, virtually all screen-oriented software that's worth anything, including text editors, word processors, and spreadsheets, goes directly to display memory rather than through the BIOS.

Finally, I'd like to point out that there's nothing to prevent you from moving display memory to A000:0000 in order to modify the upper 8 text pages, then moving display memory back to B800:0000 so that you can use the BIOS to access the lower 8 text pages. You can readily display the contents of the upper 8 pages while display memory is located at B800:0000; the CPU just can't read or write those pages until display memory is moved back to A000:0000. In order to see this ad-


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

mittedly unusual feature in action, set NUMBER_OF_PAGES to 16, RELOCATE_TO_A000 to 1, and RELOCATE_BACK_TO_B800 to 1 in Listing 1. This causes the program in Listing 1 to switch the display memory segment back to B800 before displaying the 16 pages; the upper 8 pages are displayed even though the CPU can't read from or write to them given the selected memory addressing.

How is it that the code in Listing 1 can display 8 text pages that the CPU can't get at? Basically, so long as you're in text mode, the VGA simply reads video data from display memory, beginning at the start address you've specified; the address at which the CPU can (or can't) address display memory is irrelevant so far as fetching video data is concerned, just as the start address is irrelevant so far as CPU addressing of display memory is concerned. Consequently, the same video data is displayed whether display memory resides in the segment at A000, B000, or B800. The only difference is that the CPU can't get at the upper 8 pages when display memory lies in the segment at B000 or B800.

**The Page Bit Reveals 16 More Text Pages**

Now we've found 16 text pages, and since 16 times 4K bytes equals 64K bytes, it would seem that that's all the text pages we can have (given that we're not going to use 128K mode). There is, however, a very interesting bit that lets us get at 16 more text pages, for a grand total of 32 text pages. That bit is the Page bit, bit 5 of the Miscellaneous Output register at port 3C2h.

The Page bit is normally 1 in text mode; this causes CPU reads and writes to access the portion of display memory that's displayed with start addresses in the range 0 through 7000h. When the Page bit is set to 0, CPU reads and writes go to the portion of display memory that's displayed with start addresses in the range 8000h through 0F000h. This portion of display memory is normally untouched in text mode because no BIOS mode supports access to it, but there's no reason not to use it. Figure 5 shows the locations of all 32 text pages, and their interaction with the start address.

The 16 additional text pages that can be accessed when the Page bit is 0 are a mirror copy of the 16 pages we've already encountered. If display memory is located in the segment at 0B000h or 0B800h, 8 text pages are accessible when the Page bit is 0 and 8 more text pages are accessible when the Page bit is 1, and the BIOS will draw text to whichever set is currently selected by the Page bit. (However, the BIOS will always program the VGA to display the cursor in the normal set of pages—that is, in the pages that the CPU accesses when the Page bit is 1.) If display memory is located at 0A000h, the 16 text pages we've already seen are accessible when the Page bit is 1, and the 16 alternate text pages are accessible in exactly the same fashion when the Page bit is 0.

Listing 2 illustrates all 32 text pages of the VGA. One interesting aspect of Listing 2 is that the same text page, page 0, is displayed while all 32 pages are filled with text, and then the Page bit is left at the same value while each of the 32 pages is displayed in turn by changing the start address. The key here is that the start address (which affects which portion of display memory is shown on the screen) is independent of the Page bit (which affects which portion of display memory CPU reads and writes go to).

The code in Listing 2 sets the Page bit to zero by reading from port 3CCh, then ANDing off bit 5 and writing the new


**SCREEN MANAGER**
For C, BASIC, FORTRAN, COBOL Programmers

**The Screen Generator ™**

The Screen Generator offers professional screen management for creation of Window, Help, Menu, and Data Entry Screens for over 20 compilers. You create screens using SG's Screen Painter. All screen text and field information is stored in compact screen libraries. All Screen Display and Data Entry functions are controlled from within your program. SG's function calls provide all the flexibility and speed you'll ever need for screen display and block mode or field level data entry. The Screen Generator includes: The Screen Painter/Print Utility, The Screen Manager, and 100 page User Guide.

*So give us a call! You'll be up and running in no time, while our 60-day Money Back Guarantee eliminates all the risk!*

**The Screen Generator ™ — New Version**

| | |
|---|---|
| C, BASIC, ASM & Turbo Pascal | $79. |
| FORTRAN or COBOL | add $20. |
| Demo Disk Available | VISA/MC |

**The West Chester Group**
P.O. Box 1304, West Chester, PA 19380
(215) 644-4206

Number **190** on the Reader Service Card

**PLOT TEXT ON ANY GRAPHICS SCREEN!!**

**YES,** we said **ANY** graphics screen, even VGA! ***Finally!*** **Xgraf** is a super set of low level assembly graphics routines that you call directly from Compiled Basic. The **Xgraf** kernel replaces Basic's confusing statements with consistent full featured calls specifically designed for the Basic programmer. **Xgraf** supports **Hercules, CGA, EGA, EEGA, VGA, and user defined screen sizes.** All modes can be accessed in real time - no recompiling!

***Advanced features extend Basic graphic capabilities!***

In addition to Basic's graphics features, **Xgraf's** ***advanced features*** include zooming, tiling, a virtual screen page, screen compression to disk, windowing, screen importing, *and much more!*

***Finally!*** **Xgraf is only $99.00 + $4.00 s&h**
(Xgraf requires the Microsoft QuickBASIC compiler)

We *specialize* in libraries and tools for Compiled Basic. Our catalog features the ***Finally!*** family of programmer's libraries and *other top flight tools.*

**KOMPUTERWERK - YOUR BASIC RESOURCE**

**KOMPUTERWERK**
851 Parkview Blvd.
Pittsburgh, PA 15215

**Call: 1-800-423-3400**
PA & AK call (412) 782-0384
VISA and MC accepted!

Number **155** on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

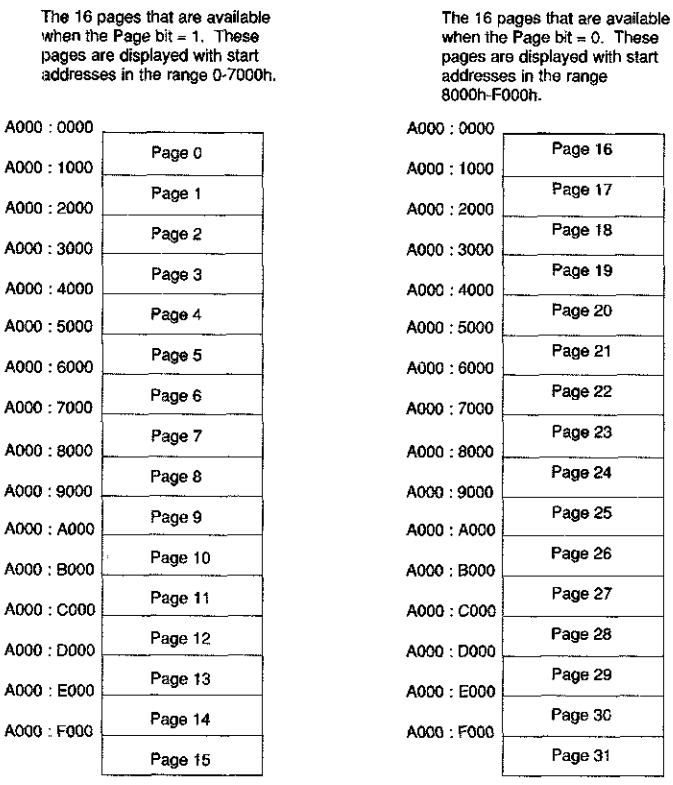



Figure 5: The 32 VGA text pages that are available when the Page bit is used.

setting to port 3C2h. On the VGA, the Miscellaneous Output register is readable only at 3CCh since 3C2h is used on reads for the Input Status Zero register. (The Input Status Zero register is supported by the EGA, so the VGA's designers presumably had to leave it in the same place on the VGA for EGA compatibility.)

On the EGA, the Miscellaneous Output register is not readable at all, and that poses a problem. The EGA requires you to know the value of the entire Miscellaneous Output register because there's no way to read the current Miscellaneous Output setting and alter only the Page bit; the problem is that it's difficult to know what that value should be. A Miscellaneous Output register setting that works on the EGA won't necessarily work on the VGA, and vice versa. Moreover, different Miscellaneous Output settings may be required on the EGA depending on the type of monitor attached, and adapters that are BIOS-level compatible with the VGA may not work with either VGA or EGA settings.

What to do? Well, it's possible to look up the Miscellaneous Output register setting in the BIOS parameter table, but even then there is a problem since the VGA's text modes are 400-scan-line modes and thus are located in a different area of the BIOS parameter table than are the EGA's 350-scan-line modes. While we could even work around that, it's not easy, and I've found BIOS parameter tables to be less than totally reliable in the past, especially on clone adapters. Yet another possible solution would be to set the Miscellaneous Output register depending on the setting of the BIOS points variable, which contains the height of the current font in pixels, but that's getting into very tricky territory. In short, manipulating the Page bit reliably on all VGAs and EGAs is no simple task.

*Continued*.

★ NOW AVAILABLE FOR C ★ QUICKBASIC ★
★ IBM BASIC/BASICA ★ TURBO PASCAL ★

The Aspen Systems Subroutine Editor (**ASE**) you can call from your programs. Design your own screen layouts - - update in several windows simultaneously. **ASE** is:

- ★ **TRANSPORTABLE** - can be configured for any keyboard and almost any computer or language under MSDOS.
- ★ **VERSATILE** - customize input for any application.
- ★ **COMPLETE** - with full screen edit capabilities and a wide variety of automatic conversions.
- ★ **EASY TO USE** - data and screen layouts described in a single map.
- ★ **FULLY DOCUMENTED** - including examples in BASIC, PASCAL, FORTRAN, C, and even COBOL.
- ★ **AFFORDABLE**-
  **ASE** is still only..... **$99**
  **DEMO** (See what ASE can do for you!)..... **$ 3**

The Aspen Systems Subroutine Package (**ASP**) - the companion package for **ASE** - with many functions unavailable or difficult to perform in high level languages. Like **ASE, ASP** is:

**TRANSPORTABLE ★ FULLY DOCUMENTED**
**EFFICIENT ★ AFFORDABLE**

**ASP** includes 150+ subroutines: conversions; sorts; string, bit, date/time functions; decimal arithmetic, and **MORE**.

**ASP** is still only..... **$130**

**ASE** and **ASP** include support and demo programs.

Prices are PPD (continental USA).
Colorado Residents add 3%.

P. O. Box 1163
Grand Junction, CO 81502
(303) 245-3262
VISA/MasterCard accepted

ASPEN SYSTEMS

CP/M and MS-DOS are trademarks of Digital Research and Microsoft, respectively.

Number **86** on the Reader Service Card

HGC TOOLS™

**Hercules™**
InColor Card
Graphics Card Plus
Graphics Card

**MODE & REGISTER CONTROL**: Routines provide flexible programming of the 6845 CRT controller chip used on the Hercules™ family of graphics cards, including ALL extension registers. Facilities efficiently control the various modes of operation of the cards in areas such as Text, RamFont,™ Graphics and Color processing.

**COMPREHENSIVE HEADER FILES**: Both C and Assembler "Include" files provided an extensive array of Register mnemonics, macro control routines and global data area control.

**EMULATION LIBRARY**: A library of graphics routines which extends the functionality of the Microsoft® QuickC™ / C5.x Graphics.LIB to the Hercules™ Cards, including full color processing on the InColor™ Card. A superset, the many functions provide additional features in support of the Hercules™ environment.

**Soft Evolutions, Inc.**
484 Lake Park Ave., #17
Oakland, CA 94610-2730
(415) 523-8247

**$79.95**
**POSTPAID**

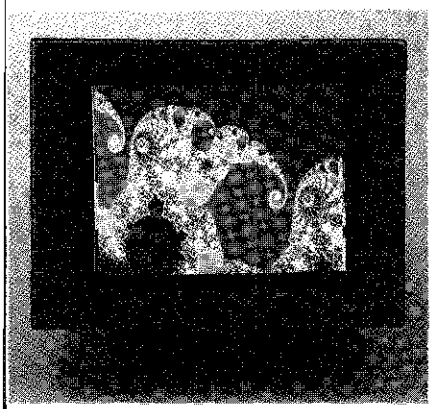
Number **177** on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

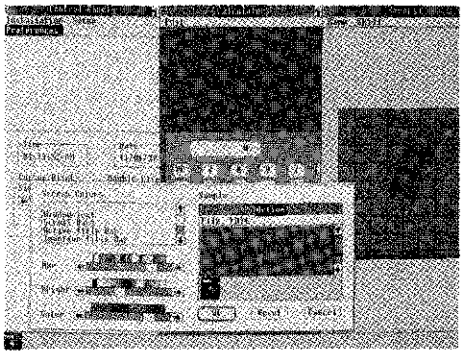
# VERY HIGH RESOLUTION

**The PC Tech COLOR and MONOCHROME video processor boards employ the TMS 34010 high performance graphics co-processor to insure the best possible video performance at reasonable prices.**



## Color 34010 Video Processor:

- **Featured on the cover of Micro Cornucopia.**
- **From 800 x 512 through 1024 x 800 resolution (depending on monitor and configuration).**
- **8 Bits per pixel for 256 simultaneous colors**
- **Hardware support for CGA/MDA emulation.**
- **PC, XT, and AT compatible**

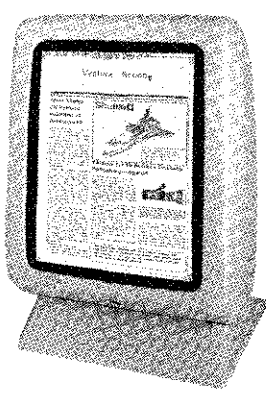


The PC Tech Color 34010 video processor is a superior 34010 native code and DGIS development tool. We support up to 4 megabytes of program (non-display) 34010 RAM as well as up to 768K bytes of display RAM. ***Compare our architecture and prices to any other intelligent graphics board. Then choose the PC Tech Color 34010 Video Processor for your development engine and your production requirements as well.***

## Color 34010 Video Processor . . . . . $1,195.00

Price includes 512K display RAM, 1024K program RAM, and utility software. Monitor not included.
Also available: DGIS, 34010 C compiler, assembler, 34010 fractal software,
additional display and program memory, and various monitor options.

# PC Tech Monochrome 34010 Video Processor and Monitor



- **736 x 1024 resolution (other options available)**
- **2 bits per pixel for 4 hardware gray shades**
- **Hardware support for CGA/MDA/Hercules emulation**
- **PC, XT, and AT compatible**
- **Full page 66 line text editing with many popular editors**
- **Excellent windows 2.0 application development system**

The graphics and bit manipulation capabilities of the TMS 34010 make the PC Tech Monochrome 34010 Video Processor 66 line full page text and graphics display faster than many 25 line systems. The video processor is available separately or with the high resolution white phosphor monitor shown above.

## Monochrome 34010 Video Sub-System . . . . . $1,295.00

Price includes Monochrome Video Processor and monitor pictured above.
Also available: DGIS, TI 34010 C compiler, TI assembler.
Monochrome 34010 Video Processor also available separately.

***Designed, Sold and Serviced By:***

PCTech

904 N. 6th St.
Lake City, MN 55041
(612) 345-4555
(612) 345-5514 (FAX)

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Number 208 on the Reader Service Card

Unbelievable!

**SOURCER™**

- **SEE HOW PROGRAMS WORK**
- **EASILY MODIFY PROGRAMS**

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Complete support for 8088 through 80286, V20/V30, 8087, and 80287 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

*On my list of programs that I simply won't do without!*
—Robert Hummel, Senior Technical Editor, PC Magazine

**SAMPLE OUTPUT**

*Fully automatic*

*Program header*

*Assembler directives*

*Determines data areas and type*

*Detailed comments*

*Simulator follows segment changes*

*Easy to read format*

```
resetprn.lst    ResetPRN v1.01                Sourcer Listing    19-Apr-88   4:05 pm      Page 1

                                PAGE  60,132
                                ;-----------------------------------------------------------------
                                ;                          RESETPRN
                                ;
                                ;       Created:   15-Apr-88
                                ;       Version:   1.01
                                ;-----------------------------------------------------------------

= 0008                          data_1e         equ     8               ; (0040:0008=378h)

                                ;-------------------------------------------------------- seg_a -----
                                seg_a           segment para public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                resetprn        proc    far
658E:0000                       start:
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.01', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 0D
658E:0011  0040                 data_2          dw      40h
658E:0013  0D 0A 52 65 73 65    data_3          db      0Dh, 0Ah, 'Reset Printer2 $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 32 20 24
658E:0025                       loc_1:
658E:0025  0E                                   push    cs
658E:0026  1F                                   pop     ds
658E:0027  BA 0013                              mov     dx,offset data_3  ; (658E:0013=0Dh)
658E:002A  B4 09                                mov     ah,9
658E:002C  CD 21                                int     21h               ; DOS Services  ah=function 09h
                                                                          ;  display char string at ds:dx
658E:002E  B4 01                                mov     ah,1
658E:0030  CD 21                                int     21h               ; DOS Services  ah=function 01h
                                                                          ;  get keybd char al, with echo
658E:0032  3C 79                                cmp     al,79h            ; 'y'
658E:0034  75 16                                jne     loc_3             ; Jump if not equal
658E:0036  8E 1E 0011                           mov     ds,data_2         ; (658E:0011=40h)
658E:003A  8B 16 0008                           mov     dx,ds:data_1e     ; (0040:0008=378h)
658E:003E  83 C2 02                             add     dx,2
658E:0041  B0 08                                mov     al,8
658E:0043  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 8, initialize printer
658E:0044  B9 8000                              mov     cx,8000h
658E:0047                       locloop_2:
658E:0047  E2 FE                                loop    locloop_2         ; Loop if cx > 0
658E:0049  B0 0C                                mov     al,0Ch
658E:004B  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 0Ch, init & strobe off
658E:004C                       loc_3:
658E:004C  B4 4C                                mov     ah,4Ch            ; 'L'
658E:004E  CD 21                                int     21h               ; DOS Services  ah=function 4Ch
                                                                          ;  terminate with al=return code
                                resetprn        endp
                                seg_a           ends

                                ;-------------------------------------------------- stack_seg_b ----
                                stack_seg_b     segment para stack
6593:0000  00C0[FF]                             db      192 dup (0FFh)
                                stack_seg_b     ends

                                                end     start
```

(Source code output and inline cross reference can also be selected)

**BIOS SOURCE**

- **CHANGE AND ADD FEATURES**
- **CLARIFY INTERFACES**

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER $99.95  BIOS Pre-Processor* $49.95  SOURCER w/BIOS Pre-Processor $139.95

USA Shipping & Handling $3; Outside USA $15; CA Residents add local sales tax 6, 6.5 or 7%; *requires SOURCER

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

**1-800-662-VCOM**
(1-800-662-8266)

V COMMUNICATIONS
3031 Tisch Way, Suite 905, Dept. J3, San Jose, CA 95128 (408) 296-4224

PS/2, AT, XT, and PC are trademarks of IBM Corp.

Number **125** on the Reader Service Card

Perhaps the simplest solution to the problem of how to program the Miscellaneous Output register is to use only the first 16 text pages. Another solution is to detect the installed adapter and handle the VGA with one set of instructions and the EGA with another set. (Listing 2 doesn't detect adapter type, but it is capable of handling either a VGA or an EGA; VGA code is assembled when ADAPTER_IS_VGA is non-zero, and EGA code is assembled when ADAPTER_IS_VGA is zero.) There may well be a more elegant solution; if you know of one, please write and tell us.

Finally, you may wonder exactly where an additional 64K bytes of memory comes from when the Page bit is flipped. The answer is detailed in the VGA section of either the Model 50/60 or the Model 80 *Technical Reference* manual, as well as in the *Display Adapter Technical Reference* supplement—if you understand memory-addressing figures and tables. If you don't understand the explanations in those manuals, don't worry; there are few cases indeed in which you need to understand the Byzantine ways in which the VGA organizes its memory, and the code in Listings 1 and 2 will suffice for almost all text-paging needs. If you really need to know more—well, the information's all there in IBM's manuals!

## In Conclusion

Once again, we've found that the VGA is much more complex and capable than we'd expect—the 4 simple text pages that were carried over from the CGA proved to be only the tip of the iceberg. The 32 text pages of the VGA are enough to support some animation, or to store many pre-made text screens without placing any demands on normal system memory.

Believe it or not, there's plenty more where that came from; the VGA is a seemingly inexhaustible source of new and useful tricks and techniques. Next time I'll probably look at saving and restoring VGA screens, but, as always, I'm open to suggestions. See you then. ♦

*Michael Abrash is a Senior Software Engineer for Orion Instruments, Inc., of Redwood City, California, where he is working on the design of advanced PC-based test instrumentation. Readers may write to Michael at P.O. Box 39051, Mountain View, CA 94039, or drop Michael a line via MCI Mail (user name MABRASH).*

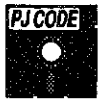
*Code follows.*

**Did You Know . . .**

The original tracking ball cursor positioning devices used a billiard cue ball (evidence of the mechanical designer's misspent youth)?—*W.G. Madison* ♦



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386. Mail order information on page 89 .

## Code from "VGA Text Paging" by Michael Abrash

1

**Listing 1**

```
;
; Program to demonstrate the 4 text pages of the CGA.
;
; To demonstrate the 8 standard text pages of the VGA/EGA,
; change NUMBER_OF_PAGES from 4 to 8.
;
; To demonstrate the 16 text pages available on the VGA/EGA
; when display memory is relocated to A000, change NUMBER_OF_PAGES
; to 16 and change RELOCATE_TO_A000 from 0 to 1.
;
; To demonstrate that the upper 8 text pages can be displayed even
; when the CPU can't access them, change NUMBER_OF_PAGES to 16,
; RELOCATE_TO_A000 to 1, and RELOCATE_BACK_TO_B800 to 1.
;
; By Michael Abrash  5/22/88
;

NUMBER_OF_PAGES         equ     4
RELOCATE_TO_A000        equ     0
RELOCATE_BACK_TO_B800   equ     0
PAGE_LENGTH             equ     1000h   ;number of bytes per page

if RELOCATE_TO_A000
DISPLAY_MEMORY_SEGMENT  equ     0a000h
else
DISPLAY_MEMORY_SEGMENT  equ     0b800h
endif

GC_INDEX                equ     3ceh
GC_MISCELLANEOUS        equ     6
CRTC_INDEX              equ     3d4h
CRTC_START_ADDRESS_HIGH equ     0ch
CRTC_START_ADDRESS_LOW  equ     0dh

stack   segment para stack 'STACK'
        db      512 dup (?)
stack   ends

code    segment para public 'CODE'
        assume  cs:code, ds:nothing
start   proc    near
;
; Set to text mode.
;
        mov     ax,3
        int     10h
;
; Relocate display memory to A000 if necessary, by reprogramming
; the Graphics Controller's Miscellaneous register.
;
if RELOCATE_TO_A000
        mov     dx,GC_INDEX
        mov     al,GC_MISCELLANEOUS
        out     dx,al
        inc     dx
        mov     al,06h
        out     dx,al
endif
;
; Point to the segment that display memory is currently
; addressed at.
;
        mov     ax,DISPLAY_MEMORY_SEGMENT
        mov     es,ax
;
; Fill each of the text screens in turn with an appropriate character.
; (Page 0 is filled with "A", page 1 with "B", and so on.)
;
        sub     ax,ax   ;start with page 0
        mov     bl,'A'  ;base letter
FillPageLoop:
        push    ax
        call    FillPage
        pop     ax
        inc     ax
        cmp     ax,NUMBER_OF_PAGES
        jb      FillPageLoop
;
; Relocate display memory back to B800 if necessary,
; by reprogramming the Graphics Controller's Miscellaneous
; register, to illustrate the principle that it's possible to
; display the upper 8 pages even when the CPU can't access them.
;
if RELOCATE_BACK_TO_B800
        mov     dx,GC_INDEX
        mov     al,GC_MISCELLANEOUS
        out     dx,al
        inc     dx
        mov     al,0Eh
        out     dx,al
endif
;
; Display each of the text pages in turn, waiting for a key
; before going to the next page.
;
        sub     ax,ax
ShowPageLoop:
        push    ax
        call    ShowPage
        call    WaitKey
        pop     ax
        inc     ax
        cmp     ax,NUMBER_OF_PAGES
        jb      ShowPageLoop
;
; Done-perform a mode set to reset all paging and clear
; the screen, then return to DOS.
;
        mov     ax,3
        int     10h
        mov     ah,4ch
        int     21h
start   endp
;
; Fills a text page with a corresponding character.
;
; Input:
;       AX = text page
;       BL = character for text page 0 (each text page
;               is filled with character BL + text page #)
;
FillPage        proc    near
        push    ax      ;save the page #
        mov     dx,PAGE_LENGTH
        mul     dx      ;calculate the start offset of the page
                        ; in the display memory segment
        mov     di,ax
        pop     ax      ;get back the page #
        add     al,bl   ;convert it to a unique letter
        mov     ah,7    ;attribute is white
        mov     cx,PAGE_LENGTH/2 ;page length as measured
                                 ; in character/attribute pairs
        cld
        rep     stosw   ;fill the page
        ret
FillPage        endp
;
; Programs the CRTC to display the desired text page.
;
; Input: AX = text page.
;
```

2

*Code continues . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Abrash Code, Continued

```
ShowPage    proc    near
            mov     dx,PAGE_LENGTH/2
            mul     dx              ;start offset of the pages as
                                    ; measured in character/attribute
                                    ; pairs (which is how the CRTC
                                    ; counts in text mode)
            mov     dx,CRTC_INDEX   ;point to the CRTC Index register
            push    ax
            mov     al,CRTC_START_ADDRESS_HIGH
            out     dx,al
            inc     dx              ;point to the CRTC Data register
            mov     al,ah
            out     dx,al           ;set the high start address byte
            dec     dx              ;point to the CRTC Index register
            mov     al,CRTC_START_ADDRESS_LOW
            out     dx,al
            inc     dx              ;point to the CRTC Data register
            pop     ax
            out     dx,al           ;set the low start address byte
            ret
ShowPage    endp
;
; Waits until a key is pressed.
;
WaitKey proc    near
            mov     ah,1
            int     16h
            jz      WaitKey
            sub     ah,ah
            int     16h             ;clear the key
            ret
WaitKey endp

code        ends
            end     start
```

**Listing 2**

```
; Program to demonstrate the 32 text pages of the VGA/EGA.
;
; To demonstrate the 32 text pages of the VGA, set
; ADAPTER_IS_VGA to 1.
;
; To demonstrate the 32 text pages of the EGA, set
; ADAPTER_IS_VGA to 0.
;
; By Michael Abrash  5/22/88
;
ADAPTER_IS_VGA          equ     0
PAGE_LENGTH             equ     1000h   ;number of bytes per page
MISCELLANEOUS_OUTPUT_W  equ     3c2h    ;Miscellaneous Output write port
MISCELLANEOUS_OUTPUT_R  equ     3cch    ;Miscellaneous Output read port
GC_INDEX                equ     3ceh
GC_MISCELLANEOUS        equ     6
CRTC_INDEX              equ     3d4h
CRTC_START_ADDRESS_HIGH equ     0ch
CRTC_START_ADDRESS_LOW  equ     0dh

stack   segment para stack 'STACK'
        db      512 dup (?)
stack   ends

code    segment para public 'CODE'
        assume  cs:code, ds:nothing
start   proc    near
;
; Set to text mode.
;
        mov     ax,3
        int     10h
;
; Relocate display memory to A000 by reprogramming
```

*Code continues . . .*


# FREE SOFTWARE

Finally, a Window Library Complete with
**C Source Code & Make Files!**

AEWINDOS works with
- Microsoft 4.0/5.0/QUICKC
- Borland TURBO C
- Lattice C 3.1/3.2

Evaluate the package for 30 days under No Obligation. If you like the package (and we're betting you will), keep it. We'll bill you for $149.95 after 30 days.

Available now for MS-DOS/PC-DOS, look for MS-OS/2 and Unix versions this summer!
Demo disk available for only $5.00.




- writes directly to video memory for *SPEED!*
- writes through the BIOS for compatibility!
- comes with a WYSIWYG window editor!
- documentation is TSR, hot-key into it!
- over 100 library functions in K&R C!

ORDER NOW FOR A FREE 30 DAY
NO-RISK TRIAL 1-800-634-5494
(499-7332 inside Colorado)

ÆWINDOS
*by* ÆSOFT

*Turboc is a trademark of Borland Int., Quick C and MS-OS/2 are trademarks of Microsoft, Lattice C is a trademark of Lattice, Unix is a trademark of AT&T Bell Labs, PC-DOS is a trademark of IBM.*

Number 220 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# "A truly important and informative conference for SOFTWARE DEVELOPERS."

—Michael Swaine
Dr. Dobb's Journal

If you are a professional software developer and can only attend one conference this coming year, don't miss Computer Language's SOFTWARE DEVELOPMENT '89 — February 14-17, 1989 in San Francisco.

## A different kind of conference - - by and for programmers.

Garry Ray of PC Week said it best. SOFTWARE DEVELOPMENT "appeal(s) to the doers of the PC business - - the people who go to the trenches every day to design, develop and debug the programs that fuel the microcomputer revolution."

At SOFTWARE DEVELOPMENT '89 you'll have three days to focus on practical techniques that will increase your programming power. You'll learn from the most important visionaries and practitioners in software development today.

People like:

Larry Constantine, P.J. Plauger, Bjarne Stroustrup, Jim Brodie, Dick Gabriel, Jon Bentley, Rich Finkelstein, Robert Ward, Jack Purdum, JimWarren and dozens more.

## Learn the latest development tools and techniques.

Imagine the chance to pick and choose from over 100 lectures and workshops on design methodologies, programming tools and utilities, object-oriented programming, C and other programming languages, artificial intelligence, graphics interface design and much more, all tailor-made for you - - the experienced software developer.

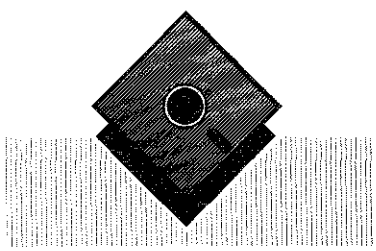
You'll have lots of time outside class to share problem-solving tips and tricks with the"best and brightest" in organized roundtable discussions, informal evening receptions and vendor parties.

And you'll be able to try out the latest programming tools from the industry's leading suppliers at the most targeted Development Products exhibit anywhere.

## Register now and save.

Cost for the four day event is only $695. But if you register now, and pay by October 14, 1988 you pay only $475 - - a $220 savings.

You can register by phone. Just call Bill Rutledge at (415) 995-2471. Or return the coupon below. DO IT TODAY.



**SOFTWARE DEVELOPMENT '89**

**San Francisco**
**February 14-17, 1989**

Produced and managed by the publishers of Computer Language, AI Expert, UNIX Review, Database Programming & Design and Embedded Systems Programming.

"It is here, and only here, that the future can be determined."

Garry Ray
—PC WEEK

Send to: Software Development '89, Miller Freeman Publications, 500 Howard Street, San Francisco, CA 94105

Number 167 on the Reader Service Card

**YES!** Please register me to attend Software Development '89 set for February 14-17 1989 on the San Francisco Peninsula. Registration fees: $695. Pay only $475 if payment is sent before 10/14/88.

❑ Bill me. ❑ Payment enclosed.

❑ VISA ❑ MC ❑ AMEX #_____ Exp._____

Signature_____

❑ Please send course information ❑ Please send exhibit information

Name _____

Title _____

Company _____

Address _____

City, State, Zip _____

Phone _____

Software Development Seminars, 500 Howard Street, San Francisco, CA 94105 PJ910

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Abrash Code, Continued

4

```
; the Graphics Controller's Miscellaneous register.
;
        mov     dx,GC_INDEX
        mov     al,GC_MISCELLANEOUS
        out     dx,al
        inc     dx
        mov     al,06h
        out     dx,al
;
; Point to display memory.
;
        mov     ax,0a000h
        mov     es,ax
;
; Fill text pages 0-15 (the pages that are accessible when

; the Page bit is 1, the setting after a mode set is performed).
; (Page 0 is filled with "A", page 1 with "B", and so on.)
;
        sub     ax,ax   ;start with page 0
        mov     bl,'A'  ;base letter for pages 0-15
FillPages0_15Loop:
        push    ax
        call    FillPage
        pop     ax
        inc     ax
        cmp     ax,16
        jb      FillPages0_15Loop
;
; Set the Page bit to 0.
;
if ADAPTER_IS_VGA
        mov     dx,MISCELLANEOUS_OUTPUT_R
        in      al,dx
        and     al,not 20h      ;Page bit (bit 5) -> 0
        mov     dx,MISCELLANEOUS_OUTPUT_W
        out     dx,al
else
        mov     dx,MISCELLANEOUS_OUTPUT_W
        mov     al,87h          ;Page bit (bit 5) -> 0
        out     dx,al
endif
;
; Now fill text pages 16-31 (the pages that are accessible when
; the Page bit is 0).
; (Page 16 is filled with "Q", page 17 with "R", and so on.)
;
        sub     ax,ax   ;start with page 0
        mov     bl,'Q'  ;base letter for pages 16-31
FillPages16_31Loop:
        push    ax
        call    FillPage
        pop     ax
        inc     ax
        cmp     ax,16
        jb      FillPages16_31Loop
;
; Display each of the text pages in turn, waiting for a key
; before going to the next page. Note that all 32 pages can
; be displayed without changing the Page bit; the Page bit
; only affects CPU accesses, not video data fetching.
;
        sub     ax,ax
ShowPageLoop:
        push    ax
        call    ShowPage
        call    WaitKey
        pop     ax
        inc     ax
        cmp     ax,32
        jb      ShowPageLoop
;
; Done-perform a mode set to reset all paging and clear
; the screen, then return to DOS.
```

5

```
;
        mov     ax,3
        int     10h
        mov     ah,4ch
        int     21h
start   endp
;
; Fills a text page with a corresponding character.
;
; Input:
;       AX = text page
;       BL = character for text page 0 (each text page
;               is filled with character BL + text page #)
;
FillPage        proc    near
        push    ax      ;save the page #
        mov     dx,PAGE_LENGTH
        mul     dx      ;calculate the start offset of the page
                        ; in the display memory segment
        mov     di,ax
        pop     ax      ;get back the page #
        add     al,bl   ;convert it to a unique letter
        mov     ah,7    ;attribute is white
        mov     cx,PAGE_LENGTH/2 ;page length as measured
                                 ; in character/attribute pairs
        cld
        rep     stosw   ;fill the page
        ret
FillPage        endp
;
; Programs the CRTC to display the desired text page.
;
; Input: AX = text page.
;
ShowPage        proc    near
        mov     dx,PAGE_LENGTH/2
        mul     dx              ;start offset of the pages as
                                ; measured in character/attribute
                                ; pairs (which is how the CRTC
                                ; counts in text mode)
        mov     dx,CRTC_INDEX   ;point to the CRTC Index register
        push    ax
        mov     al,CRTC_START_ADDRESS_HIGH
        out     dx,al
        inc     dx              ;point to the CRTC Data register
        mov     al,ah
        out     dx,al           ;set the high start address byte
        dec     dx              ;point to the CRTC Index register
        mov     al,CRTC_START_ADDRESS_LOW
        out     dx,al
        inc     dx              ;point to the CRTC Data register
        pop     ax
        out     dx,al           ;set the low start address byte
        ret
ShowPage        endp
;
; Waits until a key is pressed.
;
WaitKey proc    near
        mov     ah,1
        int     16h
        jz      WaitKey
        sub     ah,ah
        int     16h     ;clear the key
        ret
WaitKey endp

code    ends
        end     start
```

◆



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# 386 HummingBoard

# TWO PROCESSORS ARE BETTER THAN ONE

**Prices Start at just $1,595**

When it comes to 386 add-in boards, the HummingBoard is in a class by itself. It is a true coprocessor that makes productive use of the base system's original 8086, 286 or 386, running it in *parallel* with the HummingBoard's 386.

## You can adopt a whole new PC strategy.

Put your demanding applications that need more speed or memory, like 123, dBase, AutoCad, compilers, etc., on the HummingBoard while your network, Desqview, disk cache, TSR's and other applications run concurrently in the *other* 640k on the "base" system.

Our new Second-Shift™ software permits running a time consuming compile, report or other program totally in the background on the Hummingboard while you continue to use your base system for other tasks.

## Turn 640k into 900k

Another HummingBoard benefit is more memory for DOS programs, *without changing your software.* On the HummingBoard, many DOS applications can address from 700k up to 900k directly, without EMS. EMS 4.0 is supported as well.

## Blazing performance

With 16, 20 and soon 25Mhz versions, plus 32-bit cached memory for no-wait-state performance, the HummingBoard is at the top of the class for fast 386 systems.

# A.I. Architects

One Kendall Square, Cambridge, MA 02139
(617) 577-8052 FAX (617) 577-9774

Number 180 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Programming on the Variant Express

by
**Tom Swan**

*Programming with Pascal's fixed-length arrays and records reminds me of a wild train ride I once took from Mexico City to the northern border. Once the wheels started rolling, I knew it was too late to change my mind about being on board.*

Shipping a compiled Pascal program out of the station gives me a similar feeling, though not so sinking. With fixed length arrays and records—common data structures in Pascal—it's too late to change my mind about the size of an array or the number of fields in a record.

Such inflexibility can lead to wasteful programming. For example, I recently came across a directory module in a commercial toolkit that declares an array of 500 file-name strings. Now, how many directories do you suppose there are with 500 files? Not many. In fact, mine rarely have more than 20 entries or so. With 64-character strings, this program wastes an average of 30K every time it lists a directory!

Leaving room for the largest conceivable directory is a commendable goal. But there are better ways to achieve the same result than using a fixed-length array.

## Variable-length Data Structures

In an earlier article ("How Many Holes Does It Take to Fill an IBM PC?" *PJ*, Jan/Feb 1987), I described how to create variable-length, or dynamic, strings to conserve memory by eliminating unused blanks at string ends. Since then, I've expanded the idea to create general-purpose, dynamic arrays and records, structures that are proving very useful as I work on upgrading a Turbo Pascal 4.0 version of my PDBS database system, originally published in book form as *Pascal Programs for Database Management* (Hayden, 1984).

Following the popular trend, the database will have modules for drop-down menus and pop-up windows. In these modules, dynamic arrays and records efficiently store menu command lists, save text behind windows, and stuff into tight spaces such minor items as window titles and menu command strings.

Because of *PJ*'s own space limitations, though, it isn't possible to list the programming for the database menus and windows modules—3,000 lines and growing faster than bamboo shoots. Instead, in this article, I contrive a smaller, but still practical, example that shows how to use dynamic arrays and records to store graphics objects in memory with no waste. The program could serve as the beginnings of a CAD system or an object-oriented drawing program similar to the Macintosh's MacDraw.

## Allocating Space for Record Variants

Creating variable-length records in Turbo Pascal is more difficult than doing the same in standard Jensen and Wirth Pascal, which can allocate space for selected record variants. An example demonstrates the problem:

```
Str80 = String[80];
RecPtr = ^RecType;
RecType =
  RECORD CASE tag:Integer OF
    1 : ( i : Integer );
    2 : ( r : Real );
    3 : ( s : Str80 )
END;
```

**RecType** is a discriminated union. The **Integer tag** field indicates (discriminates) the kind of data the record contains. Without the **tag** field identifier (**CASE Integer OF . . .**) the record becomes a free union. The only difference between the two kinds of unions is the **tag** field, a device for identifying which of the many possible variants a variable of type **RecType** holds.

As declared here, a **RecType** variable can hold one of three variants: an integer (**i**), a real number (**r**), or a string (**s**). Because the three fields overlay the same memory space inside the record, only one of the three fields exists at any time.

There are two ways to allocate space for a **RecType** variable. The first, and most familiar, is to declare a global or local variable:

```
VAR  myRec : RecType;
```

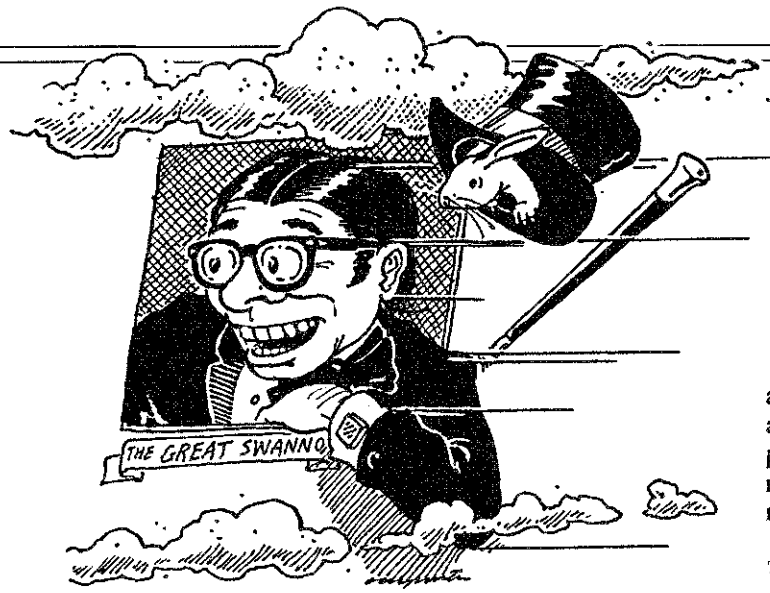
With this method, the compiler allocates space for the largest variant because it doesn't know whether you will store a 2-byte integer in **myRec.i** or an 80-character string in **myRec.s**. Obviously, if you plan to store only integers or real numbers in the record, the space allotted to the string is wasted.

The second method for declaring a **RecType** variable is to allocate space on the heap using Pascal's **New** procedure. If P is type **RecPtr**, this assigns the address of allocated memory to P:

```
New( P );
```

Unfortunately, this is no better than declaring a local or global variable. **New** must allocate space for the largest variant in case you decide to use that field. In Standard Pascal, the solution to this problem is to specify the tag constant of one


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




or more variants. For example, to allocate space for the 6-byte real number field, identified by the tag field constant 2, you can write:

```
New( P, 2 );
```

Reserving space for specific variants in Turbo Pascal requires a different approach: Call **GetMem** to allocate the number of bytes for one variant field, using the **SizeOf** function to determine the field size. Here's the Turbo Pascal equivalent of the previous standard Pascal example:

```
GetMem( P, SizeOf( Real ) );
```

Sharp readers will spot a bug here. Passing the byte size of the **Real** type does not allow for the tag field in a discriminated union. The example is correct only if P addresses a tagless free union. With an integer tag field, two more bytes are needed:

```
GetMem( P, 2 + SizeOf( Real ) );
```

***Many people have rightly criticized Pascal for requiring array sizes to be hard-wired into the source text.***

### Disposing Record Variants

Disposing record variants, returning their memory to the heap for future allocations, is the reverse process. In Standard Pascal, you simply pass the same tag field constant you originally used with **New**:

```
Dispose( P, 2 );
```

Again, Turbo Pascal doesn't recognize this standard approach. To dispose a record variant in Turbo Pascal, call **FreeMem** with the number of bytes to dispose—the same number you originally passed to **GetMem**. Add 2 to the size of the variant if the record includes a 2-byte tag field:

```
FreeMem( P, SizeOf( Real ) );
FreeMem( P, 2 + SizeOf( Real ) );
```

### Variable-length Arrays

A big plus in BASIC is the ability to dimension an array at runtime. Many people have rightly criticized Pascal for requiring array sizes to be hard-wired into the source text. Getting around this limitation is not difficult in Turbo Pascal, however, where you can add dynamic arrays that expand to a calculated size when the program runs.

There are many ways to proceed, but my favorite approach is to use a two-field record. Suppose you need an array of integer values, but, when you write the program, you don't know exactly how many array elements to declare. One solution is to create a pointer and record type along these lines:

```
TYPE
  IntPtr = ^IntRec;
  IntRec =
    RECORD
      numInts : Word;
      intArray :
        ARRAY[0..0] OF Integer
    END;
```

Type **IntPtr** is a pointer to **IntRec** variables. **IntRec** records contain two fields: **numInts**, a **Word** that indicates the maximum number of integers the array can hold; and **intArray**, the variable-length array.

From this declaration, it appears that **intArray** can hold only a single integer at **intArray[0]**. But, unknown to the compiler, when the program runs, **GetMem** allocates extra space to an **IntRec** variable, assigning the address of the expanded space to an **IntPtr** pointer. Because Turbo Pascal's range checking is normally off, you can index the array past the declared range, gaining access to the extra array elements created at runtime. The array must come last in the record to allow this expansion.

(Note: I've received letters from people confused by the ARRAY[0..0] declaration. Although this may appear kooky, it's just Pascal's way of declaring a single array element where the low and high values of the array-index range are both zero.)

After declaring the variable-length array type, the next step is to allocate space at runtime for a number of elements. In Pascal, this is best handled in a procedure:

```
PROCEDURE NewArray( n : Word;
    VAR p : IntPtr );
BEGIN
    GetMem( p, SizeOf( Word ) +
     ( n * SizeOf( Integer ) ) );
    p^.numInts := n
END;
```

In the procedure, the **GetMem** statement allocates exactly enough memory for an **IntRec** variable containing one two-byte **Word** field (**numInts**) plus an array (**intArray**) of **n** integers. The assignment to **p^.numInts** saves **n** in the record's **numInts** field, letting other statements inspect the maximum array size. Ignore this value at your own peril. Indexing **intArray** with values outside the range 0..**numInts**–1 can overwrite other items in memory.

*Continued . . .*


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

As written here, the procedure ignores the possibility that **n*SizeOf(Integer)** will be larger than the maximum memory **GetMem** can allocate, 65535 bytes. This error becomes more likely as the size of the array elements grows larger. The procedure also ignores a possible out-of-memory error, which causes **GetMem** to halt the program. These problems are easy to fix, as I'll explain in a moment. (If you can't wait, peek ahead to OBJDRAW.PAS, procedure **NewObjList**.)

With the **NewArray** procedure and the earlier TYPE declarations, a simple example shows how to create a ten-integer array at runtime, assign values to all array positions, and display the results. Examine the two different ways to limit index **i**'s range in the FOR-loops. While both methods are correct, the second is better because the subtraction is placed outside the FOR-loop, increasing execution speed.

```
VAR p : IntPtr;
    i : Integer;
BEGIN
  NewArray( 10, p );
  WITH p^ DO
  BEGIN
    FOR i := 1 TO p^.numInts DO
      intArray[i-1] := i;
    FOR i := 0 TO p^.numInts-1 DO
      Writeln( intArray[i] )
  END
END.
```

**Artful Objectives**

In graphics software, there are two fundamental types of drawing programs: object-oriented and pixel-oriented. Using the pixel variety resembles painting on canvas. When you paint a cloud over el Sol, the sun is gone for good (unless, of course, you paint a new sun). The contrasting object-oriented program resembles cartooning or special-effects filming with transparent overlays. Individual objects, stored and edited separately, combine for the final picture.

Because of the complexity of the various kinds of objects a full-blown object-oriented program stores in memory, the program makes good use of dynamic data structures, as OBJDRAW.PAS demonstrates.

To compile the program, make sure that GRAPH.TPU is on-line or installed along with CRT.TPU in TURBO.TPL, Turbo's unit library file. Change constant **GrPath** to the directory where you store your *.BGI graphics drivers. With the correct driver on disk, the program runs in most PC graphics modes, including CGA, EGA, VGA, Hercules, and others.

Before running the program, create a text file named OBJECTS.TXT with the following data:

```
5
1 0 0 350 129 4
2 150 26 222 114 15 5
2 158 3 300 79 14 1
3 160 100 75 2 0
0 200 100 15
```

The first value in this file is the number of objects (text lines) that follow. Each object starts with a number correspond-


# With Prospero PC Pascal and PC Fortran you never lose your *train...*




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ing to the comments in the enumerated **ObjType** declaration (see OBJDRAW.PAS). The values following these numbers are the data for each graphics object.

In this example, the first line indicates that 5 objects follow. The second line starts with 1, the code for an **ObjLine** object. After that come five values, corresponding to the **LineRec** record's four integer fields (**x1**=0, **y1**=0, **x2**=350, and **y2**=129), plus a single word field indicating the line's color (**lineColor**=4).

Obviously, designing graphics this way is about as much fun as mowing the lawn with a toenail clipper. A better program would let you move a pointer around the screen and display your drawing in progress. Even so, the text data listed here might represent the program's output, which you could store on disk, feed to another drawing program, or transmit via modem to a remote system.

### How OBJDRAW Operates

To understand how OBJDRAW operates, examine the **ObjRec** and **ObjList** declarations, the dynamic array and record types that use the methods outlined earlier. **ObjRec** is a discriminated union, tagged by an **ObjType** field (**objKind**). Each **ObjRec** variable stores one of the four possible objects the program knows how to draw. **ObjList** is a variable-length array record specifying the number of graphics objects (**numObjects**) plus an array of pointers (**objects**) addressing each variable-size record variant in memory.

When the program reads OBJECTS.TXT, it creates the structure illustrated in Figure 1. As you can see, there is no wasted space in this structure—each object takes only as much memory as needed. Although the overhead required for the pointers (4 bytes each) detracts a little from this efficient design, in a more complete system with dozens or even hundreds of object types and a much larger data file than shown here, the memory savings can be significant. Notice that a single pointer (**obj**) is all that's needed to access the entire affair.

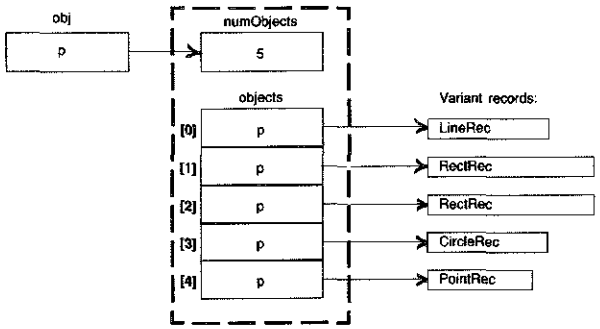



**Figure 1:** A single pointer (obj) locates the variable-length array (outlined) containing 5 pointers to graphics objects, stored as variant records in memory. A lowercase p represents a pointer variable.

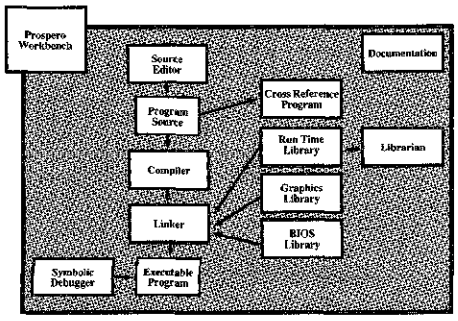
Creating this structure in memory is the job of procedure **LoadFile**, which calls **NextObject** once for each graphics object in the data file. **NextObject** calls one of four loader functions—**LoadPoint** for point objects, **LoadLine** for lines, and so on—allocating space for each record variant via **NewObj**, which calls **GetMem** as explained earlier.

Procedure **NewObjList** deals with the potential errors ignored in the earlier example. First, a local **LongInt** variable (**size**) holds the calculated number of bytes required to create an array of **n** 4-byte pointers plus 2 bytes for the **numObjects**


# ...of thought.

## Introducing the world's most productive programmer's environment: The Prospero Workbench.

Whether you program in Pascal or Fortran, you'll be more productive with the Prospero Workbench. It's a fully integrated programmer's environment for Fortran. And the world's most creative environment for Pascal.




The Prospero Workbench lets you design, write, debug, test and document a program in sequence, without interruption. So, you can never lose track of a complex variable in Fortran, or get side-tracked in Pascal.

With pop down menus and four or eight editing windows, the Prospero Workbench lets you know what's happening all the time. You can read and edit files in different windows, and easily copy text from one file to another—at the press of a key.

PC Fortran and PC Pascal include the widely praised symbolic debugger PROBE. And Prospero fully implements ISO and ANSI standards. If you prepare programs on a microcomputer, and then run them on a mainframe, you know how important that is.

Prospero Software is used by more than 20,000 programmers around the world. Prospero also provides technical support, just in case you do get side-tracked.

Call 800-327-6730 to order or to request more information, and get your programming on track with Prospero PC Fortran and PC Pascal.

PC FORTRAN $199
PC PASCAL $149

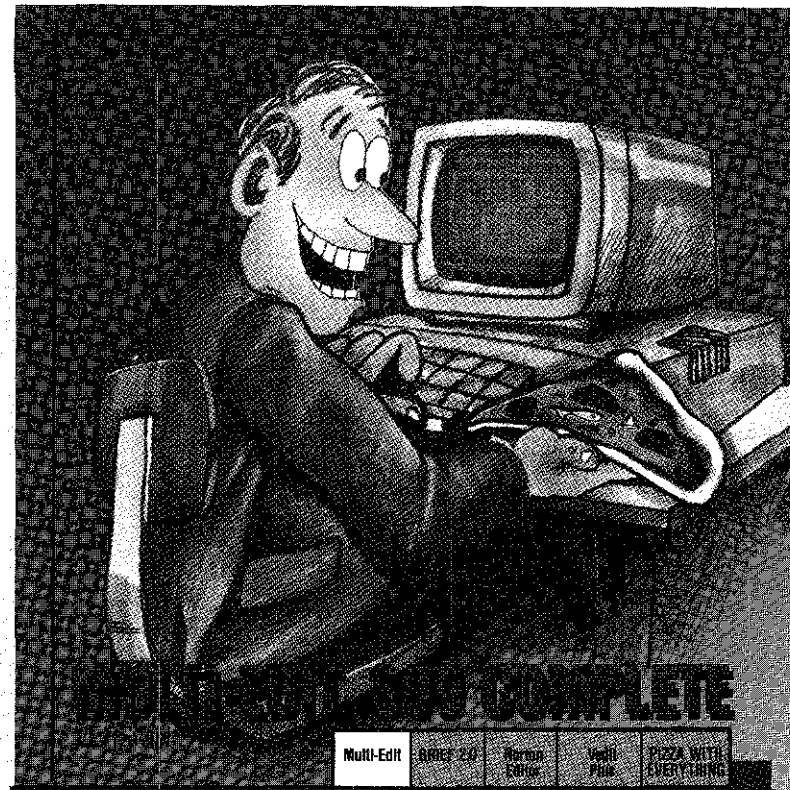
**Prospero Software**
LANGUAGES FOR MICROCOMPUTER PROFESSIONALS

**1-800-327-6730**

100 Commercial Street, Portland, Maine 04101

Number 199 on the Reader Service Card


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



# Multi-Edit vs. PIZZA

## With EVERYTHING!

- Is your editor OUT TO LUNCH?
- Does it handle ALL OF YOUR NEEDS?
- Is it flexible, programmable and reconfigurable?
- MOST IMPORTANTLY, is it EASY TO USE?

***OR WOULD YOU RATHER BE EATING PIZZA?***

## Only MULTI-EDIT tastes this good!

**Fully automatic Windowing and Virtual Memory**
- Edit multiple files regardless of physical memory size
- Easy cut-and-past between files
- View different parts of the same file

**Powerful, EASY-TO-READ high-level macro language**
- Standard language syntax
- Full access to ALL Editor functions

**Language-specific macros for C, PASCAL, BASIC and MODULA-2**
- Smart Indenting
- Smart brace/parenthesis/block checking
- Template editing
- More languages on the way

**Terrific word-processing features for all your documentation needs**
- Intelligent word-wrap
- Automatic pagination
- Full print formatting with justification, bold type, underlining and centering
- Flexible line drawing
- Even a table of contents generator

**Compile within the editor**
- Automatically positions cursor at errors
- Allocates all available memory to compiler

**Complete DOS Shell.**
- Scrollable directory listing
- Copy, Delete and Load multiple files with one command
- Background file printing

**Regular expression search and translate**

**Condensed Mode display, for easy viewing of your program structure**

**Pop-up FULL-FUNCTION Programmer's Calculator and ASCII chart**

**and MOST IMPORTANT, the BEST USER-INTERFACE ON THE MARKET!**
- Extensive context-sensitive help
- Choice of full menu system or logical function key layout
- Function keys are always labeled on screen (no guessing required!)
- Keyboard may be easily reconfigured and re-labeled
- Extensive mouse support
- Easy, automatic recording and playback of keystrokes
- Anchovies easily removed

## WHO'S NOT HOME? COMPLETE

| | Multi-Edit | BRIEF 2.0 | Norton Editor | Vedit Plus | PIZZA WITH EVERYTHING |
|---|---|---|---|---|---|
| Edit 20+ Files larger than memory | Yes | Yes | No | Yes | 2 at best |
| Powerful high level macro language | Yes | Yes | No | Yes | Human |
| Full UNDO | Yes | Yes | No | No | No |
| Visual marking of blocks | Yes | Yes | Yes | No | Looks Good |
| Line, stream and column blocks | Yes | Yes | No | No | Uses Knife |
| Auto-mark file save | Yes | Yes | No | No | No |
| Online Help | Extensive | Limited | Limited | Limited | Extensive |
| Choice of keystroke commands or menu system | Yes | No | No | Yes | Menu Available |
| Function Key assignments labeled on screen (may be disabled) | Yes | No | No | No | No |
| Word processing functions | Extensive | Limited | Limited | Extra Cost | Difficult |
| Complete DOS shell | Yes | No | No | No | Does Dishes |
| Pop-up Programmer's Calculator and ASCII Table | Yes | No | No | No ASCII | No |
| Unlimited "Undo" Keystrokes across | Yes | No | No | Yes | Knows all Edit often |
| Allocates all available memory to compiler when run from within editor | Yes | No | No | No | Lots of Bytes |
| Intelligent indenting, template editing and brace/parenthesis/block matching and checking for C, PASCAL, BASIC and Modula-2 | Yes | C Only | No | Limited | Limited, Intelligence |
| Flexible condensed mode display | Yes | No | Yes | No | Difficulty |
| PRICE | $99 | $195 | $89 | $189 | About $12 |

## Get Our FULLY FUNCTIONAL DEMO Copy for only $10!

**To Order, Call 24 hours a day:**
1-800-221-9280 Ext. 951
In Arizona: 1-602-968-1945
Credit Card and COD orders accepted.

***American Cybernetics***
1228 N. Stadem Dr.
Tempe, AZ 85281

Requires IBM/PC/XT/AT/PS2 or full compatible, 256K RAM, PC/MS-DOS 2.0 or later. Multi-Edit and American Cybernetics are trademarks of American Cybernetics. BRIEF is a trademark of Underware, Inc. Norton Editor is a trademark of Peter Norton Computing, Inc. Vedit is a registered trademark of CompuView Products Inc. Copyright 1987 by American Cybernetics.

***MULTI-EDIT COMBINES POWER WITH EASE OF USE LIKE NO OTHER EDITOR ON THE MARKET TODAY.***

Number 184 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

field in the **ObjList** record (the expandable array). Using a 32-bit **LongInt** variable avoids the nasty problem of integer or word overflow in multiplications such as this, allowing an IF statement to check that **size** is not zero and is not greater than **MaxWord**. If **size** is out of bounds, **NewObjList** sets **obj** to **Nil** and ends.

Dealing with the second error—that an out-of-memory condition will halt the program at the call to **GetMem**—requires installing a custom heap-error trap function, near the end of OBJDRAW.PAS. The compiler directive {$F+} switches on FAR generation, needed because this function will be called from a different code segment (the one that contains the runtime code for **GetMem** and **New**). Returning 1 as the function result tells **GetMem** and **New** to return **Nil** pointers when out of memory instead of ending the program. **NewObjList** tests for this condition after calling **GetMem**, saving **n** in **numObjects** only if the pointer is not **Nil**.

After loading all objects into memory, **ShowAllObjects** calls **ShowOneObj** for each of the four object types. Although this code seems long, it's composed of individual sections, each responsible for displaying one object. Displaying the entire screen is a simple matter of examining the record tag field and calling the appropriate procedure.

When you run the program, press keys 1 through 5 to bring a different object to the front. Notice how easy it is to locate individual objects with this data structure.

As you can see from these examples, variable-length arrays and variant records eliminate Pascal's normally rigid restrictions on specifying array and record sizes in the source text. Turbo's special procedures, **GetMem** and **FreeMem**, take the place of Standard Pascal's variant tag field constants in **New** and **Dispose** allowing records to store smaller variants efficiently. And, with range checking off, it's easy to create arrays that expand at runtime similar to the way BASIC dimensions an array.

Having worked out the details of using dynamic arrays and records, I'm finding many new uses for the techniques. At the same time, I'm saving a bunch of memory while letting my programs leave the station without the unnecessary baggage that too often travels with wasteful, fixed-length variables. ♦

*Tom Swan's newest book,* Mastering Turbo Pascal 4.0, Second Edition *(1988), is published by Howard W. Sams. Address letters to Swan Software, P.O. Box 206, Lititz PA 17543, or send MCI Mail to TSWAN.*

*Code follows.*


# Write Better Turbo 4.0 Programs...

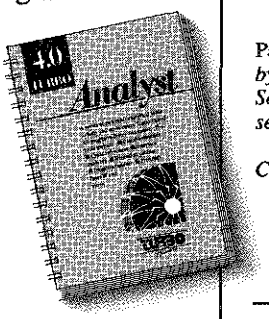
## *Or Your Money Back*

You'll write better Turbo Pascal 4.0 programs easier and faster using the powerful analytical tools of **Turbo Analyst 4.0.** You get • Pascal Formatter • Cross Referencer • Program Indexer • Program Lister • Execution Profiler, and more. Includes complete source code.

**Turbo Analyst 4.0** is the successor to the acclaimed TurboPower Utilities:

*"If you own Turbo Pascal you should own the Turbo Power Programmers Utilities, that's all there is to it."*

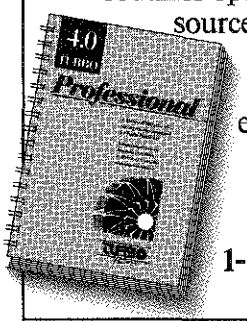
Bruce Webster, BYTE Magazine, Feb. 1986




**Turbo Analyst 4.0 is only $75.**

## A Library of Essential Routines

**Turbo Professional 4.0** is a library of more than 400 state-of-the-art routines optimized for Turbo Pascal 4.0. It includes complete source code, comprehensive documentation, and demo programs that are powerful and useful. Includes • TSR management • Menu, window, and data entry routines • BCD • Large arrays, and more.




**Turbo Professional 4.0 is only $99.**

Call toll-free for credit card orders.

**1-800-538-8157 ext. 830** (1-800-672-3470 ext. 830 in CA)

Satisfaction guaranteed or your money back within 30 days.

Fast Response Series:
■ T-DebugPLUS 4.0—Symbolic run-time debugger for Turbo 4.0, only $45. ($90 with source code)
■ Overlay Manager 4.0—Use overlays and chain in Turbo 4.0, only $45.
Call for upgrade information.

Turbo Pascal 4.0 is required. Owners of TurboPower Utilities w/o source may upgrade for $40, w/source, $25. Include your serial number. For other information call 408-438-8608. Shipping & taxes prepaid in U.S. & Canada. Elsewhere add $12 per item.

TURBO Power

TurboPower Software
P. O. Box 66747
Scotts Valley, CA 95066-0747

Number **25** on the Reader Service Card


**We Must Be Doing Something Right—or Are We?**

The International Organization for Standardization (ISO) typically adopts the national standards of its member bodies and then recommends that the other member bodies adopt the same standards.

During my tenure as Chair of the staff planning committee for the group within the American National Standards Institute that worries about computers and such, the ISO adopted the ANSI COBOL Standard.

A short time later, a colleague who had been in Russia told of seeing the Russian COBOL Standard. It was simply our own beloved ANSI Standard, with the ANSI cover replaced by the cover of the Russian standards body. The text hadn't even been translated!—*W.G. Madison* ♦



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386. Mail order information on page 89.

## Code from "Variant Express" by Tom Swan

1

```
PROGRAM ObjDraw;

{ Raw beginnings of an object-oriented graphics program,
demonstrating a practical use for variable-length data structures.
Written for Programmer's Journal by Tom Swan, Swan Software, P.O. Box
206, Lititz PA 17543. }


USES  Crt, Graph;


CONST

   FileName = 'OBJECTS.TXT';      { Graphics objects file name }
   GrPath = 'C:\TPAS4';           { Pathname to BGI drivers }
   MaxWord = 65535;               { Maximum Word value }


TYPE

   ObjType = ( {0} ObjPoint, {1} ObjLine, {2} ObjRect, {3} ObjCircle );

   PointRec =                     { Single-pixel points }
      RECORD
         x, y : Integer;               { Location of point }
         pointColor : Word             { Color of point }
      END;

   LineRec =                      { Straight lines }
      RECORD
         x1, y1, x2, y2 : Integer;     { Line endpoints }
         lineColor : Word              { Line color }
      END;

   RectRec =                      { Squares and rectangles }
      RECORD
         x1, y1, x2, y2 : Integer;     { Rectangle corners }
         lineColor : Word;             { Outline color }
         fillColor : Word              { Interior color (0=none) }
      END;

   CircleRec =                    { Circles }
      RECORD
         x, y : Integer;               { Center coordinate }
         radius : Word;                { Length of radius in pixels }
         lineColor : Word;             { Outline color }
         fillColor : Word              { Interior color (0=none) }
      END;

   ObjPtr = ^ObjRec;              { Pointer to various graphics objects }
   ObjRec =
      RECORD CASE objKind : ObjType OF
         ObjPoint    : ( onePoint  : PointRec  );
         ObjLine     : ( oneLine   : LineRec   );
         ObjRect     : ( oneRect   : RectRec   );
         ObjCircle   : ( oneCircle : CircleRec )
      END;

   ObjListPtr = ^ObjList;         { Pointer to list of graphics objects }
   ObjList =
      RECORD
         numObjects : Word;                    { Number of objects }
```

2

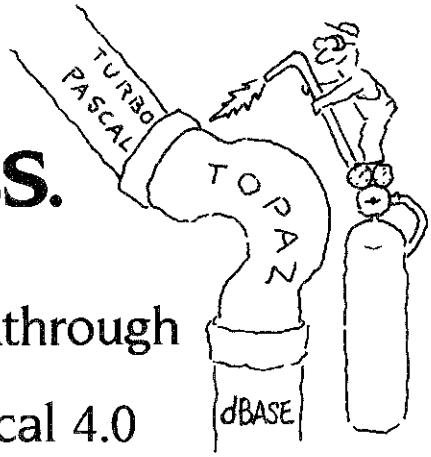
```
         objects : ARRAY[ 0 .. 0 ] OF ObjPtr    { Variable-length array }
      END;


VAR

   obj : ObjListPtr;     { Pointer to list of objects }


PROCEDURE NewObjList( n : Word; VAR obj : ObjListPtr );

{ Return pointer obj to an ObjList record large enough to hold n
ObjPtr pointers in the obj^.objects array field.  If obj=Nil on
return, then 1) n=0; or 2) bytes requested > MaxWord; or 3) enough
memory for n items is not available. }

VAR   size : LongInt;    { Number of bytes to allocate }

BEGIN
   size := SizeOf( Word ) + ( LongInt(n) * SizeOf( ObjRec ) );
   IF ( size = 0 ) OR ( size > MaxWord ) THEN obj := Nil ELSE
   BEGIN
      GetMem( obj, size );     { Out-of-memory error sets obj to Nil }
      IF obj <> Nil
         THEN obj^.numObjects := n
   END { if }
END; { NewObjList }


PROCEDURE NewObj( n : Word; VAR obj : ObjPtr );

{ Return pointer obj to an ObjRec record large enough to hold n
bytes plus the record tag field.  Out-of-memory error returns
obj = Nil. }

BEGIN
   GetMem( obj, n + SizeOf( ObjType ) )
END; { NewObj }


FUNCTION NextObject( VAR f : Text ) : ObjPtr;

{ Read next object data from disk, creating an ObjRec record large
enough to hold the data, and returning the address of this record
as the function result.  Out-of-memory error returns Nil. }

VAR   objCode : Word;       { Object code number (from data file) }
      p : ObjPtr;           { Temporary single object pointer }

   FUNCTION LoadPoint : ObjPtr;
   { Load one point object }
   BEGIN
      NewObj( SizeOf( PointRec ), p );     { Allocate memory }
      IF p <> Nil THEN WITH p^.onePoint DO
         Read( f, x, y, pointColor );      { Read data }
      LoadPoint := p                       { Return function result }
   END; { LoadPoint }

   FUNCTION LoadLine : ObjPtr;
   { Load one line object }
   BEGIN
      NewObj( SizeOf( LineRec ), p );
      IF p <> Nil THEN WITH p^.oneLine DO
         Read( f, x1, y1, x2, y2, lineColor );
      LoadLine := p
   END; { LoadLine }
```

*Code continues . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# YOU'LL LOVE THESE UTILITIES.

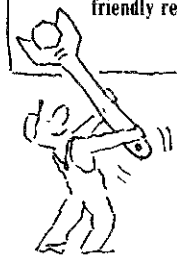



## SAYWHAT?! The lightning-fast screen generator

## TOPAZ. The breakthrough toolkit for Turbo Pascal 4.0

# WE GUARANTEE IT!

**IRON CLAD MONEY-BACK GUARANTEE.**
**If you aren't completely delighted with Saywhat or Topaz, return them within 30 days for a prompt, friendly refund.**



It doesn't matter which language you program in. With Saywhat, you can build beautiful, elaborate, colorful screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can all be displayed with as little as one line of code in *any* language. Batch files, too.

With Saywhat, what you see is *exactly* what you get. And response time is snappy and crisp, the way you like it. That means screens pop up instantly, whenever and wherever you want them.

Whether you're a novice programmer longing for simplicity, or a seasoned professional searching for higher productivity, you owe it to yourself to check out Saywhat. For starters, it will let you build your own elegant, moving-bar menus into any screen. (They work like magic in any application, with just one line of code!) You can also combine your screens into extremely powerful screen libraries. And Saywhat's remarkable VIDPOP utility gives all languages running under PC/MS-DOS, a whole new set of flexible screen handling commands. Languages like dBASE, Pascal, BASIC, C, Modula-2, FORTRAN, and COBOL. Saywhat works with all the dBASE compilers, too!

With Saywhat we also include a bunch of terrific utilities, sample screens, sample programs, and outstanding technical support, all at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

If you'd like to combine the raw power and speed of Turbo Pascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. You see, Topaz (our brand new collection of units for Turbo Pascal 4.0) was specially created to let you enjoy the best of *both* worlds. The result? You can create truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you create outstanding, polished programs, fast. Think of it. With Topaz you can write Pascal code using SAYs and GETs, PICTURE and RANGE clauses, then SELECT and USE databases (real dBASE databases!), SKIP through records, APPEND data, and lots more.

In fact, we've emulated nearly one hundred *actual dBASE commands and functions*, and even added *new* commands and functions to enhance the dBASE syntax! All you have to do is declare Topaz's units in your source code and you're up and running!

The bottom line? Topaz makes writing sophisticated Pascal applications a snap. Data entry and data base applications come together with a minimum of code and they'll always be easy to read and maintain.

Topaz comes with a free code generator that automatically writes all the Pascal code you need to maintain a dBASE file with full-screen editing. Plus outstanding technical support, at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed). **$49.95**

**ORDER NOW. YOU RISK NOTHING.** Thousands of satisfied users have already ordered from us. Why not call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

**SPECIAL LIMITED-TIME OFFER!** Buy Saywhat?! and Topaz together for just $85 (plus $5 shipping & handling). That's a savings of almost $15.

To order: Call toll-free

**800-468-9273**

In California: **800-231-7849**
International: 415-571-5019

The Research Group
88 South Linden Ave.
South San Francisco, CA 94080

☑ **YES.** I want to try:

**Saywhat?!** your lightning-fast screen generator, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

**Topaz,** your programmer's toolkit for Turbo Pascal 4.0, so send _____ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

☐ **YES.** I want to take advantage of your special offer! Send me _____ copies of both Saywhat?! and Topaz at $85 per pair (plus $5 shipping & handling). That's a savings of almost $15.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

☐ Check enclosed ☐ Ship C.O.D. ☐ Credit card

# _____ Exp. date _____ Signature _____

**THE RESEARCH GROUP**

Number 198 on the Reader Service Card

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Swan Code, Continued

3

```
   FUNCTION LoadRect : ObjPtr;
   { Load one rectangle object }
   BEGIN
      NewObj( SizeOf( RectRec ), p );
      IF p <> Nil THEN WITH p^.oneRect DO
         Read( f, x1, y1, x2, y2, lineColor, fillColor );
      LoadRect := p
   END; { LoadRect }

   FUNCTION LoadCircle : ObjPtr;
   { Load one circle object }
   BEGIN
      NewObj( SizeOf( CircleRec ), p );
      IF p <> Nil THEN WITH p^.oneCircle DO
         Read( f, x, y, radius, lineColor, fillColor );
      LoadCircle := p
   END; { LoadCircle }

BEGIN
   Read( f, objCode );                  { Read object code number }
   CASE ObjType( objCode ) OF
      ObjPoint    : p := LoadPoint;     { Read point data }
      ObjLine     : p := LoadLine;      { Read line data }
      ObjRect     : p := LoadRect;      { Read rectangle data }
      ObjCircle   : p := LoadCircle     { Read circle data }
   END; { case }
   IF p <> Nil
      THEN p^.objKind := ObjType( objCode );   { Save code as tag field }
   NextObject := p                      { Return function result }
END; { NextObject }


PROCEDURE LoadFile( VAR obj : ObjListPtr );

{ Read graphics objects from a disk file.  Halts on errors. }

VAR   f : Text;                 { Text file variable }
      n : Word;                 { Number of objects }
      i : Word;                 { For-loop control variable }

BEGIN
   Assign( f, FileName );       { Assign file name to file variable }
   Reset( f );                  { Open file for input }
   Read( f, n );                { Read number of objects }
   NewObjList( n, obj );        { Create array to hold list of n objects }
   IF obj = Nil THEN            { Check for bad n or short memory }
   BEGIN
      Writeln;
      Writeln( 'Cannot allocate space for ', n, ' objects' );
      Writeln( 'Memory available = ', MemAvail );
      Halt(1)
   END; { if }
   FOR i := 1 TO n DO           { Read n objects from disk }
      obj^.objects[i-1]         { Read next object and }
         := NextObject( f );    {  assign to variable-length array }
   Close( f )
END; { LoadFile }


PROCEDURE ShowOneObj( obj : ObjListPtr; n : Word );

{ Display object number n in object list addressed by obj pointer.
Assumes obj is not Nil.  Ignores any Nil pointers in obj^.objects
array. }

VAR   p : ObjPtr;     { Holds copy of obj^.objects[n] }
```

4

```
   PROCEDURE ShowPoint( VAR onePoint : PointRec );
   { Display point object }
   BEGIN
      WITH onePoint DO
         PutPixel( x, y, pointColor )
   END; { ShowPoint }

   PROCEDURE ShowLine( VAR oneLine : LineRec );
   { Display Line object }
   BEGIN
      WITH oneLine DO
      BEGIN
         SetColor( lineColor );
         Line( x1, y1, x2, y2 )
      END { with }
   END; { ShowLine }

   PROCEDURE ShowRect( VAR oneRect : RectRec );
   { Display Rect object }
   BEGIN
      WITH oneRect DO
      BEGIN
         IF fillColor > 0 THEN
         BEGIN
            SetFillStyle( SolidFill, fillColor );
            Bar( x1, y1, x2, y2 )
         END; { if }
         SetColor( lineColor );
         Rectangle( x1, y1, x2, y2 )
      END { with }
   END; { ShowRect }

   PROCEDURE ShowCircle( VAR oneCircle : CircleRec );
   { Display Circle object }
   BEGIN
      WITH oneCircle DO
      BEGIN
         SetColor( lineColor );
         Circle( x, y, radius );
         IF fillColor > 0 THEN
         BEGIN
            SetFillStyle( SolidFill, fillColor );
            FloodFill( x, y, lineColor )
         END { if }
      END { with }
   END; { ShowCircle }

BEGIN
   WITH obj^ DO
   IF ( 0 <= n ) AND ( n < numObjects ) THEN
   BEGIN
      p := objects[n];
      IF p <> Nil THEN WITH p^ DO
      CASE objKind OF
         ObjPoint    : ShowPoint( onePoint );
         ObjLine     : ShowLine( oneLine );
         ObjRect     : ShowRect( oneRect );
         ObjCircle   : ShowCircle( oneCircle )
      END { case }
   END { if }
END; { ShowOneObj }


PROCEDURE ShowAllObjects( obj : ObjListPtr );

{ Display all objects addressed by object list pointer obj.  Assumes
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
5

that obj is not Nil. }

VAR   i : Word;          { For-loop control variable }

BEGIN
   FOR i := 1 TO obj^.numObjects DO
      ShowOneObj( obj, i - 1 );
END; { ShowAllObjects }


PROCEDURE DoGraphics( obj : ObjListPtr );

{ Initialize graphics screen and display objects addressed by obj. }

VAR   grDriver, grMode, grError : Integer;   { BGI graphics variables }
      ch : Char;              { Holds keypresses }

BEGIN
   grDriver := Detect;
   InitGraph( grDriver, grMode, grPath );
   grError := GraphResult;
   IF grError <> grOk
      THEN
         Writeln( 'Graphics error : ', GraphErrorMsg( grError ) )
      ELSE
         BEGIN
            ShowAllObjects( obj );
            REPEAT
               ch := ReadKey;
               ShowOneObj( obj, ( Ord(ch) - Ord('0') ) - 1 )
            UNTIL ch = Chr(27);
            CloseGraph
         END { else }
END; { DoGraphics }


{ The following custom heap-error trap function lets GetMem and New
return Nil pointers if memory allocation requests fail due to
insufficient memory. }

{$F+}    { Switch on far-procedure generation }
FUNCTION HeapErrorTrap( size : Word ) : Integer;
BEGIN
   HeapErrorTrap := 1      { New & GetMem: return Nil if out-of-memory }
END; { HeapErrorTrap }
{$F-}    { Switch off far-procedure generation }


BEGIN
   HeapError := @HeapErrorTrap;  { Assign custom heap-error trap address }
   Writeln;
   Writeln( 'Welcome to ObjDraw' );
   Writeln;
   Writeln( 'Reads data from file ', FileName );
   Writeln( 'Press digit keys to bring objects to the front' );
   Writeln( 'Press Esc to quit' );
   Writeln;
   Write( 'Press Enter to begin...' );
   Readln;
   LoadFile( obj );              { Load objects from disk }
   DoGraphics( obj )             { Display objects }
END.
```


# TURBO SOURCE SEARCH

TSS is a programmer's SIG that currently has 240+ disks representing 80MB of archived files!! TSS specializes in locating publicly accessible files in Turbo Pascal, BASIC, QBASIC, C, QuickC, ASM and Modula-2. We offer these files by mail as well as on-line.

TSS-BBS (617-545-9131 within the PC-Pursuit area) is an outlet for the programming code from *Programmer's Journal*. All files in the TSS SIG are freely accessible through the BBS to members and non-members. Members have increased time and access. A central library for publicly accessible programming utilities is a valuable resource, and we are certain that programmers recognizing that will support our efforts by becoming members.

TSS is also a BBS-by-mail. All files maintained by TSS-BBS are available to members for as low as $2.50/disk. All files on all disks are archived to maximize the number of bytes per disk. A yearly membership to TSS, $40.00, entitles you to purchase disks from the library for $4.00/disk. Bulk discounts can lower the price to $2.50/disk. We also offer custom filled disks. Send $4.00 for the catalog disk or $3.00 for a printed catalog. (Members receive first catalog disk FREE.)

**TURBO SOURCE SEARCH**
P.O. Box 11, Scituate, MA 02066
***Voice: 617-545-6677***
Data: 617-545-9131, 1200/2400 BAUD, 24 hrs.
VISA/MC/Check/Money Order/non-US extra

Number **35** on the Reader Service Card

# dp_MAX

# dBase III Tools in Turbo Pascal 4.0

Complete Support for dBase III files.
DBF, NDX, DBT file & record Access.
Fully Compatible dBase III B+Tree ISAM.
Library of 100+ functions in TP4 Unit.
Allows 250+ files & indexes open at once.
LRU file caching, round robin file manager.

$$dp\_MAX = \int_{-\infty}^{+\infty} \frac{dBASE\ III\ +}{Turbo\ Pascal\ 4.0}$$

Max Software Consultants, Inc.
4101 Greenmount Avenue
Baltimore, MD 21218
(301)-323-5996

**$149**

Number **213** on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# An OS/2 Presentation Manager Application Template

by
**William S. Hall**

*The Presentation Manager (PM) is the graphics interface to Microsoft's multitasking operating system OS/2. Almost by definition, PM will be successful. Application developers especially will need to understand the basic structure of PM programs, and anyone involved in OS/2 should at the very least have some idea of PM's basic principles. To this end, we present a simple PM program you can use as the basis for more complex applications or as a vehicle for understanding this new programming environment.*

PM is a windowing system. If you have experience with Microsoft Windows, you are already well along the way to understanding programming in PM. Nevertheless, there are many differences between Windows and PM, some obvious and others more subtle. We are going to make references to Windows analogs in the discussion that follows, but we will try not to assume previous Windows knowledge. You may find it useful, however, to read "A Digital Clock in Windows" (*PJ* 6.4) in parallel with this article.

Space and time prevent us from giving an extensive overview of PM. Rather, we are going to read through a simple program called SPMTPL (Small Model Presentation Manager TemPLate) and show you the minimum steps required to display a window. To make the program mildly interesting, we also label the icon, not with a bitmap, but with a text string. This sort of icon display has several advantages in a magazine article since the reader does not have to construct the bitmap resource. At the same time, we can show you such useful items as how to get information about fonts and how to display text.

### Generalities about SPMTPL

SPMTPL is a natural outgrowth of SMLTPL, a similar Windows template we developed for the article "A Digital Clock in Windows" (*PJ* 6.4), which can be obtained on the issue disk or from the PJ bulletin boards. Consisting of six files, our PM template is very easy to use as the starting point for another program. For example, the first six letters in the name of each file are the name of the basic program. Thus, SPMTPL is the make file, SPMTPL.DEF is the definitions file, SPMTPL.H is the header, SPMTPL.RC contains the resource descriptors, and SPMTPL.C is the main C program. To name other modules, two additional letters are used to suggest functionality; in the template, SPMTPLNT.C is the iNiTialization module. Other possible module names are xxxxxxFN.C for a file of FuNctions called by the main window procedure and xxxxxxMS.C for a segment containing MiScellaneous operations only occasionally needed.

To use the template as the basis for another program, say MYPROG, first copy all the template files to the MYPROG directory and give the command REN SPMTPL*.* MYPROG*.*. Now edit each file, replacing the string **spmtpl** with **myprog**. In the .DEF and .RC files, you will also need to make some additional changes such as the description string, icon label string, and window title. Make the program and test it. You are now ready to add your own code.

A feature of PM (and Windows) programs is the use of several segments of code that can be loaded and moved by the operating system as needed. For moderate sized programs, the medium model containing several code and one data segments is ideal. The small model can also be used in this way if segments are named and individually described in the definitions file. Care must be taken, however, not to make calls to C library functions from segments other than the default code segment _TEXT. In the template, we have a named initialization segment _INIT marked as discardable and containing code executed only upon startup. Since the initialization code is about the same size as that needed for normal operations, the savings in run-time memory is obvious.

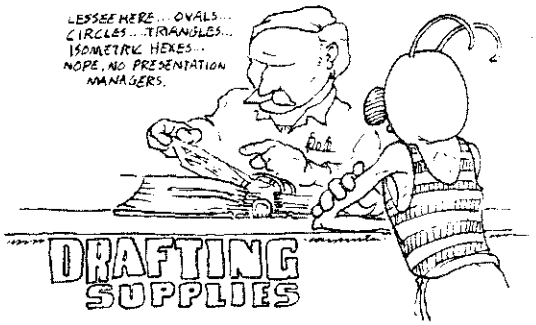
### Software

The Presentation Manager Software Development Kit (PMSDK) and the Microsoft 5.1 C compiler are necessary for PM development. Currently the PMSDK is very expensive and available only as part of the OS/2 development package. The original price of the OS/2 development package was $3000, but this may have changed recently. The C compiler can be obtained at discount for around $270.

The state of PM is also interim, so functionality is reduced and the documentation is quite far behind the current implementation. Here is where a basic knowledge of Windows is helpful. Even though *every* function in PM differs in name and functionality from Windows' functions, you can often draw on Windows experience to help you find your way. The most helpful manual along these lines is the *Conversion Guide*; I used it extensively to convert the Windows template to PM. In addition, PMSDK comes with sample source code and a set of excellent notes and examples by Charles Petzold (see the references at the end of this article). Finally, you really need to print out the various header files such as **os2def.h**, **pmwin.h**, **pmgpi.h**, etc., and keep them handy in order to know what the real arguments to PM functions are. The documentation is simply too much out of date; function parameters are often wrong, and data structure names and types have changed. Nevertheless, you still have to refer to the manuals for details about parameters. So it's a matter of jumping constantly among at least three sources to write a few lines of code. As time goes by, this situation will im-



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED




prove; it will *not* be a permanent feature of PM (we hope).

### The Environment

The basic machine must be 80286-based since you are now working in the OS/2 environment. Even though my personal machine is very fast in Windows, it is already too slow for PM. Working with PM on my 6 mHz, no-wait-state Zenith Z-241 reminds me very much of dealing with Windows on an IBM XT. As PM develops, the speed of the display will increase, but it appears that a 12 mHz 80286 or an 80386 machine may be the minimum configuration needed to do useful work.

The programming environment is also still awkward. Although the compiler and linker run under OS/2, my favorite editor (Final Word) and many of the tools I use do not. So I edit and compile in the compatibility (3X-) box then switch to PM to test the program. The switch is made a bit easier by a dual monitor EGA/MDA configuration. After booting OS/2 into PM on the EGA screen, I use the PM filing program to get to the source directory. Then I run DOS to enter the compatibility box and again change to the source directory. I use the command **mode mono** in the **autoexec.bat** file so that the 3X-box will use the monochrome display adapter and display. After editing and making the PM file, I tap Alt-Esc to switch the machine back to PM, where I can run the program.

### Some Differences between Windows and PM

Windows is basically a message reading system. In a non-Windows program, the operating system is queried for information; in Windows, the program waits for messages. When one arrives, it is processed by a windows procedure, which in form is a large case statement calling supporting functions. A window is a fairly complex object divided basically into two areas. The client area is where the program action, such as display of text and graphics, takes place. The non-client area contains objects such as menus, title bars, scroll bars, and buttons of various types. One window function is used to control both the client and the non-client areas. For the most part, actions in the non-client area are simply passed on to the default window procedure, which is managed by the system. Or, the application can act on these messages by calling other Windows functions to achieve the desired results.

PM is also a message-passing system, but the structure of the window is different. Every visible application begins by creating a frame window. The frame is itself the child of the screen, the so-called Desktop. The frame may have several children—a menu, scroll bars, control buttons, and a client window. Consequently, each has its own windows procedure managed by PM; the application normally manages only the client window procedure. If the program needs to interact with a control window, it does so by sending the control window the appropriate message rather than by calling a specific PM function.

Other differences are the PM's increased use of long integers and far pointers for data. For example, window handles as well as the last two parameters passed to a windows procedure are now unsigned longs. Rectangles and point structures have fields that are 32 rather than 16 bits in size. And, we do not want to forget to mention that rectangles now have their origin in the lower left hand rather than the upper left hand corner. In this business, it pays to have a flexible, pragmatic personality.

Now let's examine the code in some detail. We'll begin with the most important file for obtaining a general picture of a program, the **make** file.

### The Template Make File

A glance at Listing 1 shows that even a simple PM program consists of several parts. Here we have a resource segment created from SPMTPL.RC, a main module SPMTPL.C, and an initialization segment SPMTPLNT.C. Note the use of two macros, one for compiling and the other for linking with libraries. In this way it is easy to adjust the **make** file for your environment and current needs. For example, if you didn't build composite libraries when you installed your C compiler, it will be necessary to specifically name each library needed and to use the **NODefault** option. Similarly, if you intend to compile and link for debugging, several additional switches must be specified. It is much easier to make such changes in one place via macros.

Here is a brief description of the compiler switches used:

| | |
|---|---|
| -d | Display each pass (not documented) |
| -c | Make .obj files only |
| -W3 | Set highest warning level (*strongly* recommended) |
| -AS | Generate small model code |
| -G2sw | Generate 80286 code, no stack probes, windows |
| -Ox | Optimize everything |
| -Zpe | Pack structures and enable language extensions |

Use of **-G2sw**, **-c**, and **-Zp** is mandatory.

One other item of interest is the naming of the code segment for the module SPMTPLNT.C. This will cause it to be loaded as a separate segment so that it can ultimately be discarded. But, it is necessary to mention it specifically in the .DEF file as explained below.

### The Definitions File

Listing 2 shows a new element to some that will be familiar to Windows programmers, the .DEF file for SPMTPL. In addition to the name, which should agree with the executable, and a descriptor string, the .DEF file contains important information about stack and heap sizes, segments and how they should be handled by PM, a so-called stub program, names of functions in the program that are called by PM (Exports), and those functions that are imported from other programs (Imports).

In this program, we use the minimum size recommended for the program's heap and stack. Such a large size for the stack is necessary because many screen operations such as bit block transfer (**bitblt**) use the stack extensively.

*Continued . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The stub is simply a program placed at the beginning of a PM or pure OS/2 executable that runs if the program is inadvertently activated from MS-DOS. The stub used here simply announces to the user, "This program requires OS/2 Presentation Manager," and exits. However, you could conceivably have two versions of the same program, one for PM and one for DOS, with the latter as the stub for the former.

The CODE and DATA statements declare general attributes of these segments, and then specific information about the _INIT segment is given. If you want a multiply-segmented program in the small memory model, such declarations must be made. Otherwise, all segments are simply placed into _TEXT at load time and you lose the flexibility of discardable and movable segments.

Any function in your program called by PM must be exported. In this program only the main client window function falls into this category. In programs using callback functions and dialog boxes, these handlers must also be exported.

Similarly, your program may want to access functions in that class of executables known as dynamic link libraries. Since your program makes contact with such libraries only at runtime, you must import any such functions by a corresponding statement in the .DEF file to resolve externals at link time. Learning to use libraries is an art unto itself; in this program, no imports are needed.

**The Header file**

SPMTPL.H contains constant definitions for the strings in the resource file as well as declarations of certain global variables. We'll discuss resources when we look at the .RC file, so let's concentrate on the variables.

The first item of interest is the anchor block handle, **hAB**. Charles Petzold has described **hAB** very aptly as a pass to the swimming pool. It is the first item obtained from PM and the last item to be destroyed when exiting. **hAB** is used by many Windows functions. If you are familiar with Windows, you'll realize that **hAB** is perhaps the PM object most like the instance handle.

Next, a PM program reads messages, so it needs a handle to a message queue. This is the second item obtained; the handle is destroyed upon exit just before the anchor block handle is destroyed.

We have mentioned that the basic window is a frame, so a handle is required for this item as well. And, finally, the frame will usually contain a client window; **hWndMain** serves as the client window's handle.

We have also mentioned that not all features of PM are currently functional. One of these is **WinQueryWindowPos**. Certain bits in a data structure that this function fills can be used to ascertain if the window is minimized or maximized or neither. Unfortunately, these bits are not yet being set properly. So, to tell if we are iconic, we simply get the current rectangle and compare it with the size of an icon. The icon size itself is obtained during initialization and stored in **xIconsize** and **yIconsize**. Although this procedure is a bit kludgy, it is instructive since it demonstrates how certain system parameters can be read.

In this program, we want to display a short name when the program is iconic in place of having a bit-mapped icon. In Windows, if you want to place text at the top and left of the window, you simply write to position (0,0). In PM, writing to (0,0) puts the text at the bottom of the window. To write text at the top, it is necessary to know the size of the rectangle and subtract the height of the font used. In this program, we have to obtain basic font metrics and store them in **CharWidth** and **CharHeight**. Only the latter is actually used, however.

*Continued . . .*


*"If you need source code, make sure your wallet is wide open or get VITAMIN C."*
*- Computer Language, June, 1987*

Vitamin C

It's good for your system!

- Professional C function library
- 30 day money back guarantee
- Multiple bullet proof windows
- Easy full screen data entry
- Unlimited data validation
- Context sensitive help manager
- Menus like Lotus and Mac
- Programmable keyboard handler
- Text editor routines
- No royalties or runtime fees
- Library source included FREE
- Free technical support
- Free BBS at (214)418-0059
- Supports all major compilers including Microsoft 5.0
- VCScreen code generator too!
- UNIX version avaialable

Fast, flexible, versatile, reliable. Vitamin C delivers the vital combination that software professionals demand. Highly efficient, professionally crafted C code provides lightning fast displays required by today's window intensive programs.

High level functions provide maximum productivity and require little supporting code. Plus, Vitamin C's versatile, open ended design is full of hooks so you can intercept and plug-in special handlers to customize or add features to most routines.

VCScreen, our screen painter / code generator speeds your development even more! Simply draw your input forms using our interactive design editor and generate perfect C source code ready to compile and link with the Vitamin C library.

Vitamin C ..... $225
*Includes all source code FREE! For IBM PC, XT, AT, PS/2 and true compatibles. Specify compiler when ordering.*

VCScreen ..... $99.95
*For IBM PC, XT, AT, PS/2 and true compatibles. Requires Vitamin C.*

**ORDER NOW!**

We ship UPS. Please include $3 for ground, $6 for 2-day air, $20 for overnight, or $30 if outside the U.S. Texas residents add 7 1/4 % sales tax. All funds MUST be in U.S. dollars drawn on a U.S. bank. Visa & MasterCard accepted.

**(214)416-6447**

creative PROGRAMMING

Box 112097 • Carrollton, Texas 75011

Windows • Data Entry • Menus • Help • Text Editing
Plus... All Source Code FREE!

Contact this Advertiser directly




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Programmers: Go home early!

**Now Supports** Quick BASIC 4.0, Turbo Pascal 4.0

C, BASIC, Pascal, dBASE®, FORTRAN and Modula-2 programmers: be more productive by clarifying and documenting your source code.

**"Occasionally, a utility comes along that makes a programmer's life much easier. SOURCE PRINT is such a program. It contributes to the programmer's job by organizing code into a legible format and by helping to organize the documentation and debugging process."**

**— PC Magazine**

Source Print and Tree Diagrammer both have easy-to-use menus with point-and-shoot file selection, and let you search for files containing a given string. For IBM PC and compatibles with 256K.

Join thousands of programmers who are working more efficiently using Source Print and Tree Diagrammer. Order these indispensable tools today. We ship immediately, and there's no risk with our 60-day money-back guarantee. **Order both and save. Only $155.00.**

**800-257-5773** Dept. T6

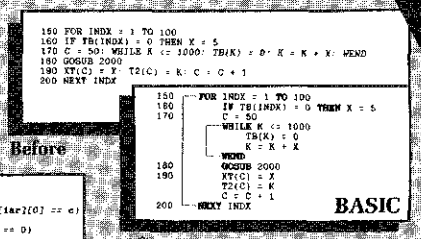
In CA call: 415-346-8325
MasterCard, VISA, American Express, COD. Add $5 for shipping/handling.

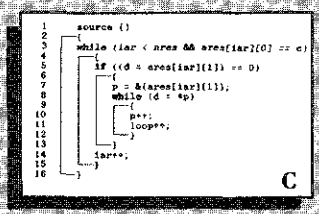
or see your local dealer!

Source Print and Tree Diagrammer are trademarks of Powerline, Inc. dBASE is a trademark of Ashton Tate. Prices subject to change without notice.

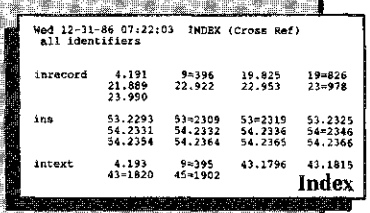
## Source Print™

***organizes your source code, simplifies debugging, and makes documentation a snap!*** It lists one or more source files with informative page headings and optional line numbers, while offering invaluable features:




**The Index** (Cross-Reference list) saves you time by showing exactly where variables are used and where functions, procedures, and routines are called.



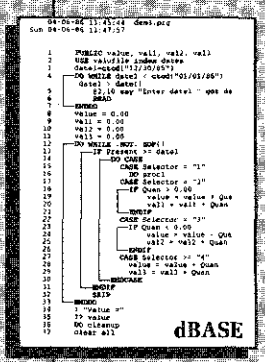

**$97.00**

Locations where new values may be assigned to variables are shown, making it easy to track down that mysterious value change.

**Structure Outlining** solves the problem of hard-to-see nested control structures by automatically drawing lines around them.

**Automatic Indentation** of source code and listings reduces your editing time and ensures indentation accuracy.

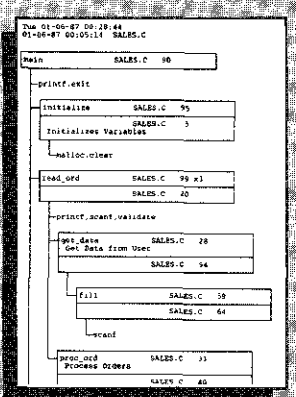
**Plus . . .** Source Print generates a table of contents listing functions and procedures. Keywords can be printed in boldface on most printers. Multi-statement BASIC lines can be split for readability. Functions and procedures can be drawn by name from one or more source files to form a new file.




## Tree Diagrammer™

**shows your program's overall organization at a glance.** Ordinary program listings merely display *functions, procedures, and subroutines* sequentially, but do not display the relationships between these routines. Our revolutionary new Tree Diagrammer automatically creates an "organization chart" of your program showing the hierarchy of calls to functions, procedures, and subroutines. Recursive calls are indicated and designated comments in the source code will appear on the chart.

Tree Diagrammer helps you organize your program more logically. And you'll be amazed at how easy it is to debug when you see how your routines interact.



**$77.00**

Number **52** on the Reader Service Card.

**Powerline, Inc.** 2531 Baker Street, San Francisco, CA 94123 415-346-8325

YES! Rush me ☐ Source Print @ $97 ☐ Tree Diagrammer @ $77
☐ Both $155. Ship/Handling $5. For CA add 6% tax Total

Name
Company
Address
City State Zip
☐ Check enclosed ☐ VISA ☐ MasterCard ☐ American Express
Card # Exp. Date
Signature Phone # T6

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

This program uses three strings in several places. They are stored in the resources and loaded into the string variables during initialization.

Finally, three function declarations shared across the two code modules are made. Note the new format you can use in Version 5.1 of the compiler. Not only are the argument types specified, but the variables themselves can also be used. You have also probably noticed that for the most part we are not using the old style function format:

```
foo(c,n)
char c;
int n;
{ ... }
```

but rather the new one:

```
foo(char c, int n)
{ ... }
```

After writing the function, it is now very easy to copy the line **foo(char c, int n)**, add a semicolon to the end, and place it in the file at the appropriate place to have a function declaration.

You have also probably seen the many new type definitions being used. You must read **os2def.h** to learn them all. Some are obvious, however—**hAB** for handle to an anchor block, ULONG for unsigned long, etc. Both OS/2 and Windows use an interesting naming method generally credited to Charles Simonyi and called the Hungarian convention (see the references at the end of this article). The naming method is very descriptive and has many advantages in this environment.

### The Resource File

SPMTPL.RC simply contains the three strings needed in the program. The strings are identified by constants and loaded when needed. Most PM programs have more complicated resource files. A resource is an item such as a menu, dialog box, string, icon, or font. Each is described using the appropriate syntax in the .RC source file, which is compiled by the resource compiler into a format that later can be attached to the program's executable.

Resources are objects that interact with the user, and consequently, they are precisely the items that must change if the program is "localized," that is, ported to another language. The isolation of resources into one module makes such internationalization considerations very convenient. In particular, resources can be changed and added back to the executable without recompiling the entire source.

### The Main Program

Perhaps it is just Windows experience, but a PM program source seems less intimidating than the corresponding one in Windows.

At the top of SPMTPL.C are the minimum include files needed for this module. All OS/2 programs must include **os2.h**. The **os2.h** file references other include files including **pm.h**. If **pm.h** sees that INCL_PM is defined, all definitions, function prototypes, and constants needed for PM are made available from other PMSDK header files by including them automatically. The programmer can sometimes reduce compile time by making additional definitions that restrict the categories of items included and hence reduce the size of the symbol table.

Below the declarations of two local functions is the entry point to the program. Unlike Windows programs, which use the entry **WinMain** and have special parameters, PM programs simply use **main**. Note the language extension **cdecl**, which declares **main** to be a C function using the usual convention of pushing arguments from right to left


NEW VERSION 2.0

MICROSOFT, TURBO AND MIX POWER C PROGRAMMERS...
C WINDOWS TOOLKIT™ PUTS YOU IN CHARGE OF VIDEO
*and* DATA-ENTRY

NEW FEATURES

- *Comprehensive data-entry and verification procedures with scanf( ) type syntax.*
- *Predefined data-entry functions for dates, SSNs, telephone numbers, currency amounts with commas, time, scientific notation, etc.*
- *Turn any window into an editor with your choice of keyboard mapping (Wordstar™ by default).*
- *Virtual windows up to available memory.*
- *Save windows to disk.*
- *Nested windows.*
- *Create and edit VGA/MCGA/EGA/Hercules fonts of any size.*

WINDOWING FUNCTIONS
- Create pop-up windows
- Create pull-down menus
- Create spreadsheet menus
- Create context-sensitive help screens
- Store windows in RAM or on disk
- Modify windows out of view
- Use 8 different types of exploding windows
- Scroll and move windows

SYSTEM CONTROL
- Device-independent output
- Supports VGA/MCGA/EGA/Hercules/CGA/MDA
- Detect the types of video adapter installed
- Switch between adapters
- Detect ANSI.SYS
- Control the size and position of the cursor
- Detect the enhanced keyboard
- Disable the video signal
- Delay program execution to microsecond resolution

VGA/EGA SUPPORT
- Use all 64 VGA/EGA colors
- Use VGA 29/43/50 line modes
- Use EGA 43-line mode
- Use 2 fonts simultaneously
- Design and edit custom fonts with FONTEDIT,™ our font editor (included)
- Smooth scroll the screen
- Smooth pan the screen

FAST SCREEN I/O
- Write formatted output (like **print( )**)
- Get snow-free output on the CGA
- Scroll the screen
- Use direct screen writes or the BIOS

HERCULES SUPPORT
- Create and edit your own Ramfonts™
- Detect the presence of a Hercules Card
- Detect Ramfont™ support
- Load and store Ramfonts™ to RAM or disk
- Switch between modes

**Full source code included — No run-time royalties.**
Includes 230 page manual — 30-day Money-Back Guarantee

Magna Carta SOFTWARE

P.O. Box 475594
Garland, TX 75047-5594

Requires: IBM PC, XT,AT, PS/2 or compatible.
Supports Microsoft C 5.0/5.1/Quick C, Borland Turbo C 1.5, Mix Power C

CALL (214) 226-6909

Only $99.95
(Texas Residents add 8% sales tax.)

Number **212** on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# C CODE FOR THE PC

*source code, of course*

| | | |
|---|---|---|
| *NEW!* | MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers) | $500 |
| | Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation) | $400 |
| | PforC or PforCe++ (COM, database, windows, file, user interface, DOS & CRT) | $345 |
| | CQL Query System (SQL retrievals plus windows) | $325 |
| | GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy) | $325 |
| | Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC) | $300 |
| *NEW!* | Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file) | $250 |
| | PC Curses (Aspen, Software, System V compatible, extensive documentation) | $250 |
| *LESS!* | Greenleaf Data Windows (windows, menus, data entry, interactive form design) | $220 |
| | Vitamin C (MacWindows) | $200 |
| | Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF) | $175 |
| | TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX) | $170 |
| | Essential resident C (TSRify C programs, DOS shared libraries) | $165 |
| | Greenleaf Functions (296 useful C functions, all DOS services) | $160 |
| | Essential C Utility Library (400 useful C functions) | $160 |
| | Essential Communications Library (C functions for RS-232-based communication systems) | $160 |
| | WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper) | $155 |
| | OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS) | $150 |
| | ME Version 2.0 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75) | $140 |
| | Turbo G Graphics Library (all popular adapters, hidden line removal) | $135 |
| | PC Curses Package (full Berkeley 4.3, menu and data entry examples) | $120 |
| | CBTree (B+tree ISAM driver, multiple variable-length keys) | $115 |
| | Minix Operating System (U**x-like operating system, includes manual) | $105 |
| | PC/IP (CMU/MIT TCP/IP implementation for PCs) | $100 |
| | B-Tree Library & ISAM Driver (file system utilities by Softfocus) | $100 |
| | The Profiler (program execution profile tool) | $100 |
| | Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.) | $100 |
| | Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.) | $100 |
| *NEW!* | TurboGeometry (library of routines for computational geometry) | $90 |
| *NEW!* | QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library) | $90 |
| | Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells | $80 |
| | C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules) | $80 |
| | JATE Async Terminal Emulator (includes file transfer and menu subsystem) | $80 |
| | MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores) | $80 |
| | WKS Library Version 1.03 (C program interface to Lotus 1-2-3 program & files) | $80 |
| | Professional C Windows (lean & mean window and keyboard handler) | $70 |
| *NEW!* | lp (flexible printer driver; most popular printers supported) | $65 |
| | Quincy (interactive C interpreter) | $60 |
| | EZ_ASM (assembly language macros bridging C and MASM) | $60 |
| | PTree (parse tree management) | $60 |
| *NEW!* | MicroFirm Toolkit (28 Unixesque utilities for MS-DOS) | $50 |
| | HELP! (pop-up help system builder) | $50 |
| | Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card) | $50 |
| | Make (macros, all languages, built-in rules) | $50 |
| | Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps) | $50 |
| | Coder's Prolog (inference engine for use with C programs) | $45 |
| *NEW!* | Virtual Memory System (least recently used swapping) | $40 |
| | C-Notes (pop-up help for C programmers ... add your own notes) | $40 |
| | Biggerstaff's System Tools (multi-tasking window manager kit) | $40 |
| | PC-XINU (Comer's XINU operating system for PC) | $35 |
| | CLIPS (rule-based expert system generator, Version 4.1) | $35 |
| | Tiny Curses (Berkeley curses package) | $35 |
| | TELE Kernel or TELE Windows (Ken Berry's multi-tasking kernel & window package) | $30 |
| *NEW!* | SP (spelling checker with dictionary and maintenance tools) | $30 |
| | Clisp (Lisp interpreter with extensive internals documentation) | $30 |
| | Translate Rules to C (YACC-like function generator for rule-based systems) | $30 |
| | 6-Pack of Editors (six public domain editors for use, study & hacking) | $30 |
| | Crunch Pack (14 file compression & expansion programs) | $30 |
| | ICON (string and list processing language, Version 7) | $25 |
| *NEW!* | FLEX (fast lexical analyzer generator; new, improved LEX) | $25 |
| | LEX (lexical analyzer generator; an oldie but a goodie) | $25 |
| | Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor) | $25 |
| | AutoTrace (program tracer and memory trasher catcher) | $25 |
| *NEW!* | Arrays for C (macro package to ease handling of arrays) | $25 |
| *NEW!* | OOPS (collection of handy C++ classes by Keith Gorlen of NIH) | $20 |
| | C Compiler Torture Test (checks a C compiler against K & R) | $20 |
| | Benchmark Package (C compiler, PC hardware, and Unix system) | $20 |
| | TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller) | $20 |
| | A68 (68000 cross-assembler) | $20 |
| | List-Pac (C functions for lists, stacks, and queues) | $20 |
| | XLT Macro Processor (general purpose text translator) | $20 |
| | C/reativity (Eliza-based notetaker) | $15 |
| | **Data** | |
| | WordCruncher (text retrieval & document analysis program) | $275 |
| | DNA Sequences (GenBank 52.0 including fast similarity search program) | $150 |
| | Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program) | $60 |
| | Webster's Second Dictionary (234,932 words) | $60 |
| | U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points) | $35 |
| | The World Digitized (100,000 longitude/latitude of world country boundaries) | $30 |
| | KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format) | $30 |
| | USNO Floppy Almanac (high-precision moon, sun, planet & star positions) | $20 |
| | NBS Hershey Fonts (1,377 stroke characters in 14 fonts) | $15 |
| | U. S. Map (15,701 points of state boundaries) | $15 |


*The Austin Code Works*
*11100 Leafwood Lane*
*Austin, Texas 78750-3409 USA*

*acw!info@uunet.uu.net*

*Voice: (512) 258-0785*
*BBS: (512) 258-8831*
*FAX: (512) 258-1342*


**Free surface shipping on prepaid orders  For delivery in Texas add 7%  MasterCard/VISA**


Number 30 on the Reader Service Card.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

QEdit *Advanced*

the "quick editor"

By SEMWARE

QEdit is one of the smallest, fastest, most easy to use multi-file editors available for IBM Compatible Computers with many features to enhance your productivity.

- *Extremely Fast*
- *Completely Configurable (Including Keyboard!)*
- *Easy to Use Macro Facility*
- *Optional "Pop Down" Menu System*
- *Recover Deleted Text*
- *Edit Dozens of Files Simultaneously*
- *Utilize up to 8 Windows to View Files*

QEdit *Advanced* requires an IBM PC/XT/AT or Compatible, IBM PS/2 30-80, 128KB of available memory. Please specify 5.25" or 3.5" diskette.

$54.95 MASTERCARD & VISA ACCEPTED

Add $3.00 Shipping & Handling. Georgia Residents Include 4% Sales Tax.

**To Order:**
SEMWARE, 730 Elk Cove Court
Kennesaw, Georgia 30144
**Phone Orders:** (404) 428-6416

Number 148 on the Reader Service Card

(necessary in a language where functions can have a variable number of arguments) and where the caller restores the stack. However, most PM functions use the Pascal calling convention in which the arguments are pushed from left to right and the callee adjusts the stack on return. (Since functions are often called from more than one place, the stack adjustment code will appear only once if the Pascal convention is used.)

The program then calls a local initialization function where the anchor block and message queue handles are obtained and the frame window is registered and created. Then, as in Windows, the program enters into its message reading loop. The program remains in the loop until **WinGetMsg** returns zero, an action triggered by receipt of a WM_CLOSE message. At this point, the frame window, message queue, and anchor block are destroyed and the program exits.

A message is a data structure whose fields include a handle to the affected window, the message itself, and two long parameters. The interpretation of the parameters depends on the message. For example, if the message is WM_CREATE, the two parameters contain pointers to data; whereas if a WM_CLOSE message is received, neither parameter is used.

An application must request messages not only to retrieve them but also to allow other applications to execute. As in Windows, while a program is waiting for a message to be returned, other applications can run. Once a message has been obtained, the application then requests PM to call its window procedure via the **WinDispatchMsg** function. Before the message is dispatched, however, the message fields can be altered although this is not usually done.

Messages are sent to an application to indicate a variety of conditions and to supply information. Keyboard input, mouse movement, state changes, and requests to redraw the window are common message generators. Often the message is not processed directly by the application but simply passed on to the default window procedure. In this program, we look at only a few.

When a window is created, a WM_CREATE message is sent. We use it as an opportunity to get some system information by calling **WndCreate**, located in the initialization segment. We'll look more closely at this function later.

If the user closes the application, a WM_CLOSE message is received. This is the opportunity for the program to do any clean-up before exiting. Here we simply place a WM_QUIT message in our own message queue; that is, we send a message to ourselves with **WinPostMsg**. When this message is retrieved, the **WinGetMsg** call returns zero and the program exits.

Every application must know how to redraw its client area. For example, a window may have a dialog box drawn over it. When the box is removed, the rectangular area where the dialog box lay must be repaired. So the program may receive a WM_PAINT message to fix the region affected. This is but one of the many reasons paint messages are sent. An application can even send such a message to itself to indicate that the data for the window has changed and that the screen must be updated. It is common for applications not to draw at the time new information is received, but rather to invalidate the affected portion of the window and hence generate a paint message. The screen is then updated when the next WM_PAINT message is processed.

In the template, we have nothing to do but display a string when the window is iconic. However, even text display is a drawing operation, and before such an operation can take place, a handle to a Presentation Space (PS) must be obtained using **WinBeginPaint**. The PS contains a certain amount of default information that can be altered before drawing begins. During the time the PS is being held, an application can change such attributes as the font, mapping mode, and colors. The PS is returned to PM by a call to **WinEndPaint** after drawing is completed.

PM will maintain the client's background with the default brush if TRUE is returned to the WM_ERASEBACKGROUND message. Otherwise, the application must paint its own background and return FALSE.

The main module ends with two functions. The first, **MainWndPaint**, examines the state of the window and if iconic, draws the string SPTL. Note that since rectangles now have an origin at the bottom left, we must perform some computations in order to place the string at the top of the icon area. And, as we explained earlier, since **WinQueryWindowPos** is not yet fully operational, we can know if the window is iconic only by comparing its width and height with the icon size. So, we have written



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Programmer's Tools

from The Software Family

## C

| | |
|---|---|
| *Borland* Turbo C v1.5 | $68 |
| *Blaise* C Tools Plus/5.0 | $95 |
| *Blaise* Turbo C Tools | $95 |
| *Blaise* Power Screen | $95 |
| *Creative* Vitamin C | $159 |
| *Creative* VC Screen | $129 |
| *Microsoft* C v5.1 | $279 |
| *Microsoft* Quick C | $68 |
| *Zortech* C++ | $99 |

## Turbo Pascal

| | |
|---|---|
| *Borland* Turbo Pascal v4 | $68 |
| *Blaise* Turbo Power Tools | $95 |
| *Blaise* Power Screen | $95 |
| *Turbo Power* T-Debug Plus | $39 |
| *Turbo Power* Turbo Professional 4.0 | $79 |
| *Turbo Power* Turbo Analyst 4.0 | $59 |

## Object Oriented

| | |
|---|---|
| *Digitalk* Smalltalk/286 New! | $164 |
| *Guideline* C++ | $269 |
| *Ithaca* HOOPS<br>*Object Oriented Graphics Database* | $499 |
| *Whitewater* ACTOR | $449 |
| *Zortech* C++<br>*Optimizing C++ Compiler* | $99 |

## Other Languages & Tools

| | |
|---|---|
| *Clarion* Professional Developer | $595 |
| *Matrix* Layout | $129 |
| *Microsoft* Macro Assembler | $98 |
| *Mortice Kern* MKS Toolkit v3 | $145 |
| Periscope I v4 | $379 |
| Periscope II v4 | $139 |
| Periscope III v4 | $1079 |
| *SLR* Optasm | $110 |

## Free Catalog & Newsletter

We carry hundreds of products from dozens of publishers. Call or write for your free subscription to our catalog and newsletter.

## Returns

The Software Family wants you to pay only for the tools you really use. If you order anything featured in this ad (or most of the products in our catalog) you can open and use the product for up to 30 days and still return it if it doesn't meet your needs.

## Screens

**Power Screen** *by Blaise* is a comprehensive screen management system for C, Turbo Pascal *and* Quick Basic. Programs can read data in block or field mode with all editing, conversion & cursor control handled by Power Screen. Can call your procedures before each field. **$95**

## 386 Hardware

**Hauppauge 386XT** replaces your XT motherboard to provide true 80386 performance and maximum compatibility. 16Mhz CPU, 1MB zero wait state. **$1590**

**386AT** **$1690**

## The Software Family

**800-443-7176**

(415)957-0111

Visa • MasterCard
COD • Check

- Most orders shipped within 24 hours
- Free shipping on orders over $100

**The Software Family**
649 Mission Street, Dept PJ98
San Francisco, CA 94105

## Visit Our Store!

**The Software Engineering Store**
575 Mission Street, 6th Floor
San Francisco
Mon-Fri 10-5

## SQL

**XQL** *by Novell* interface to Btrieve supports both embedded SQL for programs and interactive SQL for users. XQL provides sophisticated data manipulation features which allow you to access data by field name, move forwards and backwards through the database, compute fields, and even work with composite records from joined files. **$599**

## Debugging

*New version!* **Periscope II** *v4 by Periscope*: Periscope tells you where your program is hanging up, even if you don't know where to set breakpoints or have better things to do with your time than stare at traces! Whenever you detect a problem, just press Periscope's hardware NMI button, load your map files and start debugging. Periscope provides a comprehensive set of source level debugging functions including access to Microsoft C local variables and support for Microsoft Windows. For all popular languages and compilers. **$139**

## Technical Support

The Software Family carefully evaluates features, quality and support before offering a product to our customers. All of the products featured in this ad are backed by excellent technical support from the publisher — in many cases directly from the program developers. All are published by companies with reasonable update policies and with a track record for responding to customer requests for both bug fixes and new features. We at The Software Family further back the products with our own technical support. We are willing and able to intercede with the publisher if your needs aren't satisfied or to research problems caused by the interaction of products we recommend. We can also provide quotes for custom modification of source-licensed products, in-depth technical consultation or custom-developed products.

Prices subject to change without notice. ©1988 TSF All Rights Reserved.

Number 98 on the Reader Service Card.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

a simple Boolean function, **WindowIsIconic**, to do the job, and this concludes the discussion of the main module.

### The Initialization Code

SPMTPLNT.C contains code needed only during start-up. It lives in its own discardable segment and has two far entry points, **InitProgram** and **WndCreate**.

**InitProgram** does not use **argv** and **argc**, but they have been passed to this function simply because we are building a template and a more complex program may want to have this information. After the anchor block and message queue handles have been obtained, the window class is registered and the frame window is created. Along the way, the class name identifier and the window title strings are obtained from the resources with **WinLoadString**. If any of these actions fails, FALSE is returned and the program exits. Otherwise, the window is shown and TRUE is returned.

Unlike in Windows, it is not necessary in PM to perform different actions if the program has been invoked when a previous instance is also running. And the class registration call is simpler. Instead of having a class data structure initialized and the pointer passed to Windows, the necessary values are simply entered as parameters to PM's **WinRegisterClass** function.

In addition to the anchor block handle, a class name, usually the application name, a pointer to the window procedure, and a class style parameter are passed to the register class function. The only bit set in our style parameter, CS_SIZEREDRAW, ensures that the entire window will be invalidated when it is resized; this style is used when the appearance of a window depends on its size. For example, a display of centered text certainly requires a repaint if the size changes.

The last parameter in **WinRegisterClass** is the size of any extra data we want to associate with the class. Here the parameter is set to zero.

After the class is registered, we create a frame rectangle of that class with **WinCreateStdWindow**. Its parent is the Desktop (the screen). The various style parameters ensure that our frame window will be visible and have a title bar, a system menu, a thick border so that the window can be sized, and maximize and minimize buttons. We have omitted the icon bit, FS_ICON, so we can draw our own.

Note the last parameter, which is a far pointer to the client window. The client window is automatically created if this far pointer is not NULL. The client window is where the basic program action takes place.

Finally, if the frame window is successfully created, it is displayed and we return TRUE to the main program. Throughout this section of code, you become aware of the differences between PM and Windows. The two different approaches to top-level windows is reflected strongly here.

The module concludes with a short function **WndCreate**, which asks for various system values. Obtaining the icon size is quite straightforward. To get font information, first a Presentation Space handle must be obtained and a FONTMETRIC structure filled. It is then easy to get the cell size for the font along with the adjustments provided by the **lExternalLeading** field to give the recommended spacing between lines of text. Note that after the PS is no longer needed, it is released back to the PS cache.

### Conclusions

As is Windows, PM is going to be a challenge to programmers. The learning curve will be steep, especially for those lacking Windows experience. At the same time, Windows programmers are going to be tripped up by PM's conceptual differences. A lot of warmed-over Windows code will continue to work, but there will be surprises enough for all of us. Nevertheless, it is satisfying to master something really hard that is at the same time going to be mainstream computing. I hope this article makes you want to learn more about this exciting new world.

#### References


Babcock, M.J. and W. S. Hall. "A Digital Clock for Windows." *Programmer's Journal* 6.4 (July/August 1988):40–59.

Petzold, C. *Programming the OS/2 Presentation Manager, Preview Edition*. Redmond, Wash.: Microsoft Press, 1988.

Simonyi, C. "Naming Conventions." Internal memo, Microsoft Corp., Redmond, Wash. ♦



*William S. Hall holds degrees from the University of Virginia, Cambridge University, Brown University, and the University of Michigan. Mr. Hall taught college mathematics and worked for three years as an associate editor of* Mathematical Review. *He is a Senior Engineer at Olivetti Advanced Technology Center in Cupertino, California, where he specializes in Windows system support software.*


*Code follows.*


What have YOU been missing?

C Programmer's Toolbox Volumes I & II

CFlow™: Determine program hierarchy, external R/T library functions, etc.;
CLint™: More rigorously check program syntax;
CPrint™: Beautify source programs to user selected formats;
CXref™: Cross reference and check symbol usage;
CritPath™: Determine a program's critical path;
PMon™: Monitor program/OS execution; and more...

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

Easy to use. Unlimited program sizes. Online documentation...

At $79.95 per volume or $130 for both with a 30 day money back guarantee, the Toolbox is simply the best value today. The Toolbox works with your existing C compiler(s) and enhances your development environment.

Call and Order Today
Visa, MasterCard Accepted

MMC AD Systems
Box 360845 Milpitas, California 95035
(408) 263-0781
"The C Tool Specialists"

Number 94 on the Reader Service card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ CODE

All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386. Mail order information on page 89.

## Code from "Presentation Manager" by William S. Hall — 1

**Listing 1 SPMTPL**

```
# SPMtpl - Presentation Manager small memory model template.
# Written by Bill Hall, Olivetti ATC.
#

# no debug compile macro
cp=cl -d -c -W3 -AS -G2sw -Ox -Zpe

# libraries
LIBS=os2 slibcp/NOD

# inference rule for resources
.rc.res :
    rc -r $*.rc

# Dependencies
spmtpl.res : spmtpl.rc spmtpl.h

spmtpl.obj : spmtpl.c spmtpl.h
    $(cp) spmtpl.c

spmtplnt.obj : spmtplnt.c spmtpl.h
    $(cp) -NT _INIT spmtplnt.c

spmtpl.exe : spmtpl.def spmtpl.res spmtpl.obj spmtplnt.obj
    link spmtpl spmtplnt,spmtpl/align:16,spmtpl/map,$(LIBS),spmtpl.def
    rc spmtpl.res
```

**Listing 2 SPMTPL.DEF**

```
NAME     Spmtpl

DESCRIPTION     'Presentation Manager Small Memory Model Template'

STUB    'OS2STUB.EXE'

HEAPSIZE  1024
STACKSIZE 4096

CODE    MOVEABLE
DATA    MOVEABLE MULTIPLE

SEGMENTS
    _INIT PRELOAD MOVEABLE DISCARDABLE

EXPORTS
    MainWndProc @1
```

**Listing 3 SPMTPL.H**

```
/*
 * Header file for the PM small model template
 *
 * Written by William S. Hall, Olivetti ATC
 *
 */

/* This trick makes variables extern in modules EXTERN is not defined */
#if !defined(EXTERN)
#define EXTERN extern
#endif

/* resource string constants */
#define IDS_APPNAME     100
#define IDS_ICON        101
#define IDS_TITLE       102

/* global variables */
EXTERN HAB hAB;                         /* anchor block handle */
EXTERN HMQ hmqMsgQ;                     /* handle to message queue */
EXTERN HWND hwndFrame;                  /* handle to window frame */
EXTERN HWND hwndMain;                   /* handle to main client window */
EXTERN ULONG xIconsize, yIconsize;      /* size of icon */
EXTERN LONG CharWidth, CharHeight;      /* height of default font */
EXTERN char szAppName[10];              /* application name string */
EXTERN char szIcon[5];                  /* icon string */
EXTERN char szTitle[30];                /* title bar string */

/* function declarations common to both code modules */
BOOL FAR InitProgram(int argc, char *argv[]);
MRESULT FAR PASCAL MainWndProc( HWND, USHORT, MPARAM, MPARAM );
void FAR WndCreate(HWND hWnd);
```

**Listing 4 SPMTPL.RC**

```
/*
 * Resource file for the PM small memory model template
 *
 * Written by William S. Hall, Olivetti ATC
 *
 */

#include "spmtpl.h"
```

*Code continues . . .*


# Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy.

*"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours . . . I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores . . ."*

JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only $25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

*Our Guarantee:* Try the Eco-C88 compiler for $99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**
Info: **1-317-255-6476**

**Ecosoft Inc.**
6413 N. College Avenue
Indianapolis, IN 46220

**ECOSOFT**

Number 8 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# vLIB

*the user interface library for C*

**vLIB** is a comprehensive, easy to use library of over 175 custom C functions for building sophisticated PC applications.

**Windows** • *overlapping, tiled, built-in window management*

**Menus** • *vertical, horizontal pulldown, popup, arrays*

**Forms** • *full screen or popup data entry with multiple field types*

**Pop Ups** • *messages, prompts, selection lists, scroll bars*

**Formatted Screen Output**

**Full Editing Input**

**High Speed Display**

*... and much, much more!*

**Give your programs a professional look! Develop applications faster!**

Compilers supported: Microsoft C and Quick C, Borland Turbo C, Lattice C, and Datalight C.

All memory models supported.

Demo disk with usable sample programs and source (includes shipping) . $5

Library with reference card and 280 page programmer's manual . . . . . $99

Library (as above) plus full source code . . . . . $149

$5 shipping and handling. California residents please add 7% sales tax.

No royalties. Site license available.

Call or write:

*Pathfinder Associates*
**291 Madrone Avenue**
**Santa Clara, CA 95051**
**(408) 984-2256**

*Visa and MasterCard accepted.*

**BBS:** (408) 246-0164 (1200/2400 N-8-1)

Number 145 on the Reader Service Card


## Hall Code, Continued



```
STRINGTABLE
BEGIN
    IDS_APPNAME "Spmtpl"                        /* applicaton name */
    IDS_ICON    "STPL"                          /* icon string */
    IDS_TITLE   "PM Small Model Template"       /* title bar */
END
```

**Listing 5 SPMTPL.C**

```
#define INCL_PM
#include <os2.h>
#include <stddef.h>
#include <string.h>

#define EXTERN                  /* all globals are declared in this module */
#include "spmtpl.h"

/* local function declarations */
void MainWndPaint(HWND hWnd, HPS hPS);
BOOL WindowIsIconic(HWND hWnd);

/* entry point for program */
int cdecl main(int argc, char *argv[])
{

    QMSG qmsg;

  /* call initialization */
    if (!InitProgram(argc, argv))
        return FALSE;

  /* fall into message loop */
    while (WinGetMsg(hAB, (PQMSG)&qmsg, (HWND)NULL, 0, 0))
        WinDispatchMsg( hAB, (PQMSG)&qmsg );

  /* program exit */
    WinDestroyWindow(hwndFrame);
    WinDestroyMsgQueue(hmqMsgQ);
    WinTerminate(hAB);
}

/* main window message loop */
MRESULT FAR PASCAL MainWndProc(HWND hWnd, USHORT msg, MPARAM mp1, MPARAM mp2)
{

    HPS hPS;            /* handle to PM display context */

    switch (msg) {

        case WM_CREATE:         /* called when main window is created */
            WndCreate(hWnd);
            break;

        case WM_CLOSE:          /* called when main window is closed */
            WinPostMsg(hWnd, WM_QUIT, 0L, 0L) ;
            break;

        case WM_PAINT:          /* called when window needs repainting */
            hPS = WinBeginPaint(hWnd, (HPS)NULL, (PRECTL)NULL);
            MainWndPaint(hWnd, hPS);
            WinEndPaint( hPS );
            break;

        case WM_ERASEBACKGROUND:        /* have PM take care of background */
            return(TRUE);
            break;

        default:                /* all other messages go here */
            return( WinDefWindowProc( hWnd, msg, mp1, mp2 ) );
            break;
    }
    return(0L);
}

void MainWndPaint(HWND hWnd, HPS hPS)
{

  /* print a string in the icon area */
    if (WindowIsIconic(hWnd)) {
        POINTL pt;
        pt.x = 0;
        pt.y = yIconsize - CharHeight;
        GpiCharStringAt( hPS, (PPOINTL)&pt, (LONG)strlen(szIcon), (PCH)szIcon);
    }
}

/*
   Return TRUE if main window is iconic.
   This kludge is needed since WinQueryWindowPos is not yet fully implemented.
*/
BOOL WindowIsIconic(HWND hWnd)
{

    RECTL cRect;

  /* get the current window client rectangle */
    WinQueryWindowRect(hWnd, (PRECTL)&cRect);

  /* compare with size when window is iconic */
    if ((cRect.xRight == xIconsize) && (cRect.yTop == yIconsize))
        return TRUE;

    return FALSE;
```

*Code continues . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# *Double Your C Programming Productivity*

## with *Instant-C*™ and *Instant-C/16M*™

**Now you can optimize your development time as well as your C code. Use the *Instant-C* professional C development environments to develop your programs in half the time, then use your favorite C compiler to optimize your code. You get the best of both worlds: fast code from your optimizing compiler and great debugging and fast development times from *Instant-C*.**

### Faster Programming

While conventional compilers are notorious for long compile and link cycles, ***Instant-C*** professional C development environments use incremental compiler technology to completely eliminate traditional compile and link delays. Typically, you only edit a small portion of your code at one time. ***Instant-C's*** and ***Instant-C/16M's*** incremental compiler recompiles only your edited or changed code, not your entire program. This approach saves you valuable development time. Large programs can be run immediately after editing. Compare the turnaround times on typical source files after a simple change.

**Time in Seconds to Recompile and Relink**

| Benchmark | *IC/16M* | Turbo-C 1.5 | QuickC 1.0 | Microsoft C 5.1 |
|---|---|---|---|---|
| 25 line function in 955 line file* | **1** | 31 | 41 | 105 |
| 62 line function in 149 line file* | **2** | 24 | 33 | 62 |

* XCONT.C (955 lines), and XLIO.C (149 lines) used for the comparisons are part of Xlisp version 1.7, by David Betz. Each test was of a change to a single function in the file. With *Instant-C* link time is always 0. Tests used 8MHz Compaq. Total size is 8800 lines. Call for detailed test results.

### Unsurpassed Debugging

***Instant-C's*** and ***Instant-C/16M's*** source-level debugging is the best available for C. ***Instant-C*** tools have far more extensive error checking capabilities than ordinary software development tools. They automatically trap errors in: invalid pointer references, computed function addresses, out of bounds array references, overstored return addresses, arithmetic exceptions, conversion errors, invalid shifts, and invalid pointer differences. Catching any one of these errors will save you hours of valuable debugging time.

With ***Instant-C*** tools you never have to change options, recompile, or relink to use any of their powerful debugging features:

- execute any C statement
- source code animation
- source backtrace
- set/clear any number of breakpoints
- step with/without calls
- display data in any format
- monitor data for changes
- force return with supplied value
- visual breakpoints

And all debugging is done in a language you already know: C.

### Faster Testing

***Instant-C*** tools make testing your code fast and easy. Changes can be tested as quickly as they're typed. Verifying a function's operation is as easy as typing a call statement. You don't need to construct test programs or recompile your code. You can edit and test your code without loading additional programs or running under a separate debugger. With ***Instant-C*** and ***Instant-C/16M*** there are no testing delays, so your code will be more thoroughly tested and verified in much less time than other tools allow.

### More Productivity, Less Waiting

***Instant-C*** and ***Instant-C/16M*** let you increase your productivity by substantially reducing the amount of time you spend waiting for your tools to work. Tasks which require separate programs and command sets in other development systems are handled transparently by ***Instant-C*** and ***Instant-C/16M***. Services such as: cross-module declaration checking, compilation, linking, and run-time error checking are integrated with the editor, compiler, and debugger to present a seamless development environment. In addition, the tools' transparency allows you to work in familiar ways. Editor keystrokes, source formatting, and checking options are all configurable.

Because ***Instant-C*** and ***Instant-C/16M*** treat your program structure as a database, they have an intelligence which is unmatched by ordinary compilers. While editing, you can display declarations of called routines, without knowing where they're declared. You can edit and save functions by name without worrying which file contains them. And source formatting is done in your style, automatically.

Why waste your valuable development time? ***Instant-C*** tools let you create, not wait.

• ***Instant-C*** version 3.0, is an incremental C compiler, editor, debugger, interpreter, interactive object code linker, syntax checker, run-time checker, with library source. It works in conjunction with your existing compiler and is compatible with your current C programs, libraries, and hardware. It accepts Microsoft C, Lattice C, and most commercial libraries. Graphics boards and non-PC hardware are also supported. --- **$ 495.**

• ***Instant-C/16M*** version 3.2, lets you break the DOS 640K barrier and write programs which can directly access the full 16MB of extended memory available on 80286 and 80386-based PCs. ***Instant-C/16M*** gives you all the productivity benefits of ***Instant-C***, plus the ability to create larger programs. Multi-megabyte programs that are too large for Codeview™ can be run immediately with ***Instant-C/16M***. --- **$ 795.**

**30 Day Money Back Guarantee.**

**The Rational Choice for Professional Software Tools.**


Rational Systems, Inc.
P.O. Box 480
Natick, MA 01760
Tel: **(508) 653-6006**
FAX: **(508) 655-2753**



Trademarks are acknowledged to Rational Systems, Inc., Microsoft Corp., Lattice, and Borland.





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

#1 C Interpreter

# C-terp

*Develop C programs with the safety, and convenience that only our highly acclaimed C interpreter can provide.*

**"C-terp is an excellent way to develop C programs. It has a good user interface and never went astray.**

*Ken Pugh, The C Users' Group*

**"C-terp comes closest to this ideal [C development environment]"**

*John Unger, BYTE*

**"... easy to use, powerful, and a time saver ... one of my favorite production tools"**

*Larry Jordan, The C Journal*

**Not just a learning tool:** Large memory model, multiple-modules, a multi-file, reconfigurable editor, automatic make, virtual memory option, shared symbols option, complete object code compatibility, links in your compiler's entire library.

**Full screen debugging:** single-step; side-step; display variables, arrays, structures; watch expressions; watch conditions; execute any C expression (even with macros); traceback; leave-function; sticky, temporary and cursor-directed breakpoints. Out-of-bounds pointers are trapped.

**Even Bill Gates** says there's nothing like an interpreter.

Gimpel Software

3207 Hogarth Lane
Collegeville PA 19426
(215)584-4261

For MS-DOS, specify compiler
**$298.00** Microsoft 4/5, Lattice, Turbo, C86, C86 Plus, Aztec, Mark Williams
**$398.00** Xenix

Order Today - MC, VISA, COD

PA residents add 6% sales tax. Outside USA add $20. C-terp is a trademark of Gimpel Software.

Contact this advertiser directly

## Hall Code, Continued



```
}
```

**Listing 6 SPMTPLNT.C**

```
#define INCL_PM
#include <os2.h>
#include <stddef.h>
#include "spmtpl.h"

BOOL FAR InitProgram(int argc, char *argv[])
{

  /* get anchor block handle and message queue handles */
    if ((hAB = WinInitialize(NULL)) == NULL)
        return FALSE;

    if ((hmqMsgQ = WinCreateMsgQueue(hAB,0)) == NULL)
        return FALSE;

  /* This string is needed to register the Window */
    WinLoadString(hAB,NULL, IDS_APPNAME, sizeof(szAppName), (PSZ)szAppName);

    if (!WinRegisterClass(hAB,                    /* anchor block handle */
                          (PCH)szAppName,         /* class name */
                          (PFNWP)MainWndProc,     /* window procedure for class */
                          CS_SIZEREDRAW,          /* class styles */
                          0))                     /* no extra data needed */
        return FALSE;

  /* This string is needed to create the frame window */
    WinLoadString(hAB, NULL, IDS_TITLE, sizeof(szTitle), (PSZ)szTitle);

    hwndFrame = WinCreateStdWindow(HWND_DESKTOP,        /* parent */
                                   WS_VISIBLE | FS_TITLEBAR   | /*frame style*/
                                   FS_SYSMENU | FS_SIZEBORDER |
                                   FS_MINMAX,
                                   (PCH)szAppName,      /* class */
                                   (PCH)szTitle,        /* window title */
                                   0L,                  /*default client style*/
                                   (HMODULE)NULL,       /* resources in .EXE */
                                   NULL,                /* frame id */
                                   (HWND FAR *)&hwndMain);
                                                        /* handle to client */

    if (hwndFrame) {
        WinShowWindow(hwndFrame, TRUE);
        return TRUE;
    }
    return FALSE;
}

/* called when client is created */
void FAR WndCreate(HWND hWnd)
{

    FONTMETRICS FM;
    HPS hPS;

  /* get the size of an icon */
    xIconsize = WinQuerySysValue(HWND_DESKTOP, SV_CXICON);
    yIconsize = WinQuerySysValue(HWND_DESKTOP, SV_CYICON);

  /* load the icon string */
    WinLoadString(hAB, NULL, IDS_ICON, (USHORT)sizeof(szIcon), (PSZ)szIcon);

  /* get the system font character metrics */
    hPS = WinGetPS(hWnd);
    GpiQueryFontMetrics(hPS, (LONG)sizeof(FONTMETRICS), &FM);
    CharHeight = FM.lMaxBaselineExt + FM.lExternalLeading;
    CharWidth = FM.lAveCharWidth;
    WinReleasePS (hPS);
}
```

♦

**Some Progress**

How long has it been since you've written a program containing a GoTo statement (other than at assembly level)?

"GoTo Statement Considered Harmful," the letter from Edsger W. Dijkstra published in the March 1968, issue of *Communications of the ACM*, is credited (not necessarily correctly, but that's another story) with starting the mass movement towards structured, "Go-Toless" programming. Dijkstra's basic thesis is pretty much taken for granted today.

It was not always so. Following are two excerpts from a response to Dijkstra's letter written by a highly respected member of the computing community:

"I was taken aback by Dijkstra's attack on the 'go to' statement. I reject the implication that the 'quality' of a program is directly correlated with a minimal use of 'go to' and similar statements . . . . I find the emotional tone of this attack as disquieting as the 'scientific' analysis. How many poor, innocent, novice programmers will feel guilty when their sinful use of 'go to' is flailed in this letter?"

And there is still a Flat Earth Society.—*W.G. Madison* ♦



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# The Proposed ANSI C Language Standard

by
**Rex Jaeschke**

*In this installment, Rex checks WATCOM's claim that Version 6 of their C compiler is "full ANSI."*

*© Rex Jaeschke, 1988. All rights reserved.*

## WATCOM C V6

The full-page ad reads "Full ANSI C Optimizing Compiler" and "Full ANSI C Run-time Library." Despite the prematurity of these claims (there is not yet an ANSI Standard), WATCOM C is the closest thing to the draft proposed Standard I have seen yet. To be fair, the small print (in one of the manuals) indicates that the product conforms to the August 1987, ANSI C Draft. So I'll review the product in terms of that claim.

The compiler kit includes both the optimizing compiler and an integrated development environment called Express C. This ANSI-conformance review is based on the compiler only—it does not include Express C.

## The Compiler Options

Like Microsoft's compiler, WATCOM's supports an option to automatically generate prototypes from existing source. This option is /v and is extremely useful when upgrading old code and in writing new code.

The only other interesting switch is **/jd**, which allows a plain **char** to be treated as signed. (This is only for existing code since WATCOM does support the new signed keyword.)

## The Preprocessor

All the new directives are supported. Perhaps the most interesting one is **#pragma**, which WATCOM has really utilized. Quite a few powerful **pragma**s are provided to improve code generation and efficiency and to provide necessary linkage with non-C environments.

WATCOM's compiler is so confident of its ANSI compatibility that it defines the predefined macro __STDC__ as 1, meaning it *is* ANSI-conforming.

My preprocessor test suite did, however, find a couple problems. The following construct is permissible under ANSI C, yet it didn't work with WATCOM:

```
#define INCFILE2 "math.h"
#include INCFILE2
```

It did work, however, when <...> was used instead of "...".

***WATCOM C is the closest thing to the draft proposed Standard I have seen yet.***

Also, a floating-point constant in a conditional compilation expression was not diagnosed as it should have been. For example, the following piece of code was accepted without error:

```
#define MAX 10
#if MAX > 5.3
```

## The C Language

I found that WATCOM supports all of the following ANSI features:

- full function prototypes and new style function definitions
- signed keyword
- **const** and **volatile** type modifiers
- unary plus operator
- **void** functions
- concatenation of adjacent string literals
- the escape sequences **\xhhh**, **\a** and **\v**
- the **long double** type
- **long double** and **float** constants
- generic pointers (**void ***)
- **%p** in **printf** for displaying pointers
- ***** **mask** in **printf** for variable field widths
- trigraphs
- initialization of auto arrays and structures
- initialization of unions
- bit-fields in unions
- U suffix on integral types
- arbitrary breaking of tokens across lines using the backslash/newline convention

The only problem I found was that **(M)(){}** wasn't recognized as a function definition. The parentheses around the function name are redundant but legitimate and seemed to confuse the compiler.

## The Runtime Library

All of the standard headers are provided, including **locale.h**. Even the new **setlocale** function is provided although, understandably, it supports only the "C" locale. This locale is essentially "USA English mode" and is the only locale a conforming implementation need have. Since WATCOM is a Canadian-based company, it seems reasonable to suppose that they might add a French-Canadian locale in the future.

Note that numerous changes and additions have been made to the standard headers in the two ANSI meetings since



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

**ANSI C Standard Activities**

The next ANSI C meeting was initially scheduled for mid-August in Cupertino, California, but an administrative problem delayed the current public review period, so the next meeting has been rescheduled for September 26–30. It will still be held at the Sunnyvale Hilton in Sunnyvale, California, and will be hosted by Tandem, Hewlett-Packard, and Borland International.

The main goal of this meeting will be to vote out a final Standard, assuming, that is, no substantial changes are required as a result of the public review period. Note that Denmark is bringing pressure to bear at the ISO level to make a substantive change for better trigraph character support. If substantive changes result at the September meeting, the Standard will be delayed by another four to six months.

**The ISO C Standard**

On June 12–13, I attended the fourth meeting of ISO/TC97/SC22/WG14, otherwise known as the ISO Working Group for the ISO C Language Standard. The meeting, held at the British Standards Institute (BSI) in London, England, was attended by representatives from The Netherlands, Denmark, the U.K., and the U.S.A. (The convener of WG14 is P.J. Plauger, president of Whitesmiths and secretary of the ANSI C X3J11 Standards Committee.)

As a result of France's threat to vote against the ISO C Standard at the previous ISO C meeting in Amsterdam (November 1987), the ANSI C committee voted to force parentheses to group in all circumstances *including* those expressions that associate and commute. Previously, compilers were permitted to ignore grouping parentheses in expressions such as **a + (b + c)** and **(a * b) * c**. As a result of this move, the extra semantics provided with the new unary plus operator were removed although unary plus itself was retained.

The only serious issue remaining to be resolved in the ISO community is that the Danish really want a better alternative to the trigraphs proposed in the current ANSI Standard. (A trigraph is a 3-character sequence of the form **??x**, which is interpreted as some other character. For example, **??<** represents an alternate spelling for **{**, **??(** for **[**, and **??=** for **#**. There are nine such sequences, and conforming implementations *must* support all of them. Therefore, the special character constant **\?** had to be added to escape the question mark character. Trigraphs, or something like them, are necessary to support C source on terminals and machines based on the ISO-646 character set.)

The next ISO C meeting is scheduled for March 1989, in Phoenix, Arizona. (Both the location and date are tentative.) It is expected to be held jointly with the ANSI X3J11 committee. *RJ* ♦

August 1987, and future releases of WATCOM C must reflect these to be conforming. The affected headers are **assert**, **limits**, **locale**, **stddef**, **stdio**, and **stdlib**.

Recently, I found a problem with *all* my C implementations that use dummy identifier names in prototypes in the standard headers, and WATCOM's is no exception. Consider the following example:

```
#define nptr 10
#include <stdlib.h>
/* which contains the fol-
 lowing declaration
 int atoi(const char *nptr);
*/
```

The identifier name **nptr** is in the domain of the programmer's name space, so it is perfectly O.K. to use it as a macro name (despite the fact that we traditionally write macro names in all uppercase.) However, in this case, while the identifier name **nptr** in the prototype cannot conflict with any other variable, function name, etc., it does conflict with the macro name—that is, it will be replaced by 10, causing a syntax error when the prototype is compiled. To see the problem clearly, you must look at the intermediate output generated by the preprocessor.

A similar problem exists if the macro is defined as follows:

```
#define nptr []
```

Admittedly, this is a rather silly macro, but it is nevertheless a valid one. Now the meaning of the prototype is quietly changed, and you will only produce an error if the actual argument passed to **atoi** has a type incompatible with that generated by the macro expansion. This kind of bug can be difficult to track since you would not normally suspect the prototype to be the problem.

If an implementation is to become truly ANSI-conforming, it must use dummy identifier names beginning with either **__** or **_X** where **X** is an uppercase letter since such names belong to the implementer's name space. Or, since the dummy names are optional, the implementation could omit them entirely.

While you can argue that this example is rather "extreme," that's the kind of test an official conformance suite will contain, and rightly so.

**The Bottom Line**

WATCOM V6 really does appear to conform to the draft ANSI Standard of August 1987. WATCOM's done the hard work, and keeping the product up to date with each draft Standard release should be easy. The documentation set is above average although there are a few rough spots here and there. (I don't particularly care for the way WATCOM has drawn the precedence table—it attempts to combine both operator precedence and associativity in one column. I suspect that if it isn't incorrect, it's at least misleading and not what we have come to know from K&R, page 49.

WATCOM has thrown down the gauntlet to the dominant C translators. Watch out, Microsoft and Borland. Of course, the competition isn't standing still—Turbo C V2 should be out by the time you read this. Actually, so too should WATCOM V6.5. ♦

*Rex Jaeschke is an independent computer consultant, writer and lecturer. He is also a member of the X3J11 C Standard's Committee, observer on the ISO C Standard's Committee, Founding Editor of the* C Journal, *and the C Language Editor of* DEC PROFESSIONAL. *Rex's new book* Portability and the C Language *will be published by Hayden in September. It is the first text on this popular topic.*

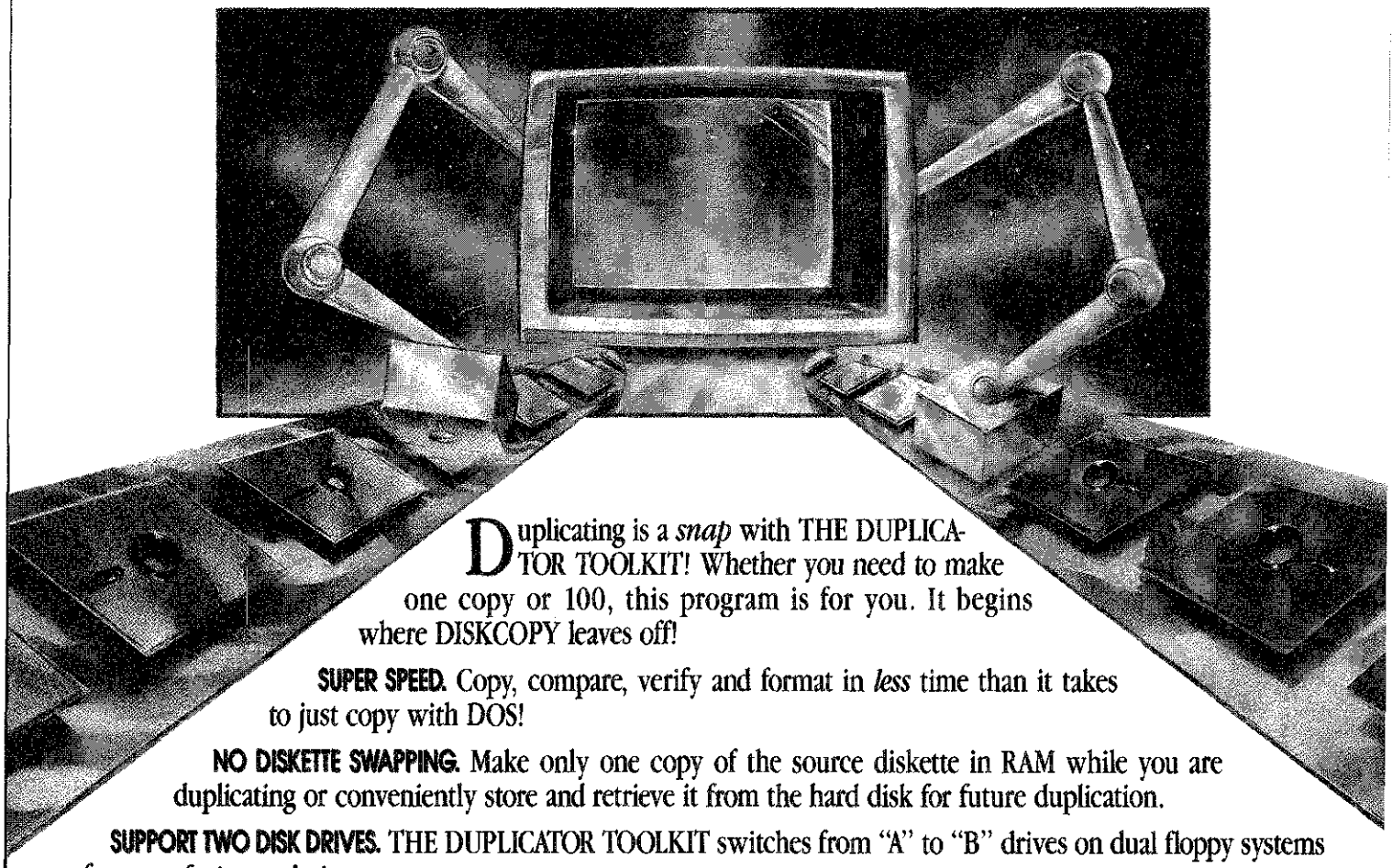
*If you would like to comment on or question any aspect of the Proposed ANSI C Standard, write to Rex Jaeschke, 2051 Swans Neck Way, Reston, VA 22091. Rex will try to address your concerns in future columns as space permits.*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

NEW!

# Turn Your PC Into A Duplicating Machine!



Duplicating is a *snap* with THE DUPLICATOR TOOLKIT! Whether you need to make one copy or 100, this program is for you. It begins where DISKCOPY leaves off!

**SUPER SPEED.** Copy, compare, verify and format in *less* time than it takes to just copy with DOS!

**NO DISKETTE SWAPPING.** Make only one copy of the source diskette in RAM while you are duplicating or conveniently store and retrieve it from the hard disk for future duplication.

**SUPPORT TWO DISK DRIVES.** THE DUPLICATOR TOOLKIT switches from "A" to "B" drives on dual floppy systems for *even faster* copies!

**EVERY COPY'S AN ORIGINAL.** Multiple verification settings let you choose the level of data checking to ensure reliable copies.

**MAKES LABELS TOO!** You can even generate labels for each diskette. Optional serialization is included.

**YOU'RE KEYSTROKES AWAY FROM PERFECT COPIES!** If even a quick guide is too much to read, this is the program for you. Easy on-screen prompts will have you duplicating in *seconds*.

Get the *power* DISKCOPY can't give you. Order today!

**MINIMUM SYSTEM REQUIREMENTS:** IBM PC, XT, AT or compatible and 256K Ram. PC or MS DOS 2.0 or higher. Hard disk recommended. Not copy-protected.

THE DUPLICATOR TOOLKIT and Copy Technologies are Trademarks of Copy Technologies.

**COPY TECHNOLOGIES**
14252 Culver Drive, Suite 323
Irvine, CA 92714

See us at
**COMDEX/Spring '88**
**May 9–12, 1988**
**Georgia World Congress Center**
**Atlanta, GA**

I've had it with DISKCOPY! turn *my* PC into a Duplicating Machine! Please send me:

_____ copies of THE DUPLICATOR TOOLKIT at $79.95 each plus $4.50 shipping ($10.00 Canada and outside U.S.). CA residents please add 6% sales tax.

NAME _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

☐ Check/Money Order ☐ Visa ☐ Master Card

Account Number _____ Exp. date _____

mail coupon to: **COPY TECHNOLOGIES**
14252 Culver Drive, Suite 323
Irvine, CA 92714

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Object Oriented OS/2

by
**Vaughn Vernon**

*As you read this article, you will find it helpful (even necessary) to reference the* C++ Programming Language *by Bjarne Stroustrup. Stroustrup's book is the definition of C++; if you will, the K&R (Kernighan and Ritchie's* The C Programming Language*) of C++. As we discuss particular facilities, some details will be left out because of space constraints. Refer to the Stroustrup book for thorough coverage.*

Object oriented programming has been getting a lot of press lately, and it's not just hype. This article focuses on C++ and OS/2 and will show you good reason to become a user of object oriented programming tools. Our discussion of OS/2 will not necessarily focus on OS/2 Presentation Manager, but rather on the kernel services and on how our software design and implementation can benefit from objects, rather than sequences of hum-drum code.

### Why C++?

Why choose C++ and not Smalltalk/V or Actor? C++ is available for OS/2. Second, speed. C++ code gets translated directly to C code and can therefore be compiled to native Intel machine code. We've all heard the arguments about the speed of C—both pros and cons.

Besides, I and a lot of other people know C, so we don't have to devote two articles to explaining some interface and language most of us have never used and may never use again. We can transfer what we know about C and benefit from the new features available in C++. I tried Smalltalk/V before I tried C++ and found it hard to learn. I think Smalltalk has a lot going for it, but I couldn't use Smalltalk everyday in my work as I do C, so I forgot as quickly as I learned.

And the very nature of properly written C++ code *forces modular programming*. Even C can leave a designer or programmer wondering how far *down* in code to place a construct, or how far removed a system dependent interface must be, and how to structure the code (leading to routines that call routines, that call routines, that . . .). C++ leaves little doubt about the proper conduct of these matters—and little room for interpretation. By use of the *class*, C++ promotes and even forces structure and readable modularity.

### Why OS/2?

Couldn't we apply object oriented concepts to MS-DOS or Unix? Indeed, nearly any application written for nearly any system interface could benefit greatly from an object oriented language. But the OS/2 system level interface is large and complex. For this major reason alone, OS/2 applications can benefit from the *forced modularity* of properly written C++ code. The complex interfaces of OS/2 can be treated as objects. And the object constructs of C++ will help make the system interface more modular.

### Benefiting from Objects

It is safe to say that most of us think in terms of objects. When we contrive a new application, we think of *programs*, *files*, *windows*, *keyboards*, *menus*, *mice*, and the *actions* each of these will take. In the process of working the idea out in our minds, we rarely think in terms of machine instructions, values in CPU registers, servicing interrupts, and efficient algorithms. These details come much later. In fact, if most of us thought of how much work will go into such an implementation up front, we might never start to code! First off, we must invent a concept.

It would be helpful in the design phase to be able to define a set of objects that express our mental pictures and ideas. That's what *object* oriented programming allows us to do. By defining a set of objects early in the design phase, we can express what it is that we want to do. We can also push details (real code) back to a later stage. (Note, however, that sometimes writing real code during design is necessary in order to find out if a particular concept will fly.)

In C++, objects are called *classes*. A class has a name, usually some data, and a set of actions that it can perform and/or that can be performed on it. The design phase of an application can be benefited by the production of a set of class names. The actions associated with each class can be defined early with almost no detail and be finished later in full detail.

A class, besides implementing an object, is used to hide detail from its user. Data and system interfaces are hidden, and the user of the class is simply presented with an object that can do certain things. With details neatly tucked away, *how* the class performs its actions can change entirely (moving from one environment to another) as long as the *things* it does don't change. This is *abstraction*.

We might reflect here on a classic example of how classes and object abstraction would have benefited OS/2 developers. For about a year, the OS/2 system call interface used full uppercase characters for routine names. In a recent release of the Software Development Kit (SDK), the entire routine interface was changed to mixed case! Besides that, system data structures (C **structs**) were changed to data type definitions (C **typedef**) and the internal member names of the data structures changed! Even for a small (less than 10K lines of code) application or tool, the necessary changes to adapt to the new routine names and data structures could be quite time consuming. In essence, the task required porting code to a new system interface.

Enter C++. Although the same changes would have to be made to C++ code as to C code, the parts of the code that would have to be changed would be much more obvious (I have personally found this to be the case). All of the C++ code that deals with a given object is located in one place. This is



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

called *encapsulation*. But most importantly, the application itself—the *things* the classes do—would not change; only the *how* part of the classes must be changed! In other words, the abstraction of an object would have kept the portions of the code that used it from knowing that their interface had changed.

Some more striking features of object oriented programming (and C++) are *inheritance* and *commonality*. Inheritance allows a new object to inherit traits of an older object, and commonality allows the new object to be used in the same way as the older object. Inheritance allows a programmer to add new features to an existing class without breaking or even modifying the existing class. You would do this by creating a new class that inherits all or part of the features of the existing class, plus some new features of its own. When a programmer defines a class from an existing class, it is called a *derived* class, and the pre-existing class is the *base*.

**Implementations of C++**

C++ is now available for OS/2. OASYS, known for its Advantage C++ for MS-DOS and Designer C++ for Unix and other systems, has produced a C++ translator for protected mode OS/2. This is an equivalent acting C++ translator, which simply operates under OS/2. We will be using the OASYS version for all of the OS/2 topics.

I received a Beta copy of the OASYS OS/2 C++. This copy came with several manuals that are useful if you are new to C++. I found the examples in this material easy to understand, and sometimes better than the Stroustrup book. The first reading I would recommend is the OASYS *Techniques Manual*. Only fourteen pages long, it is a nutshell overview of various C++ concepts. The OASYS package also comes with a *User's Guide*, *System Manual*, *Function Reference*, and a copy of the AT&T *C++ Release Notes*. In addition to providing support for the OASYS implementation of the translator, these further explain the concepts and use of C++.

Other C++ implementations are available, but not for OS/2 development. Guidelines Software has ported the AT&T C++ translator to MS-DOS. Since it is a direct port of the AT&T sources, it will translate the same C++ code under MS-DOS that the AT&T version translates under Unix (provided that the code does not support any Unix system specific interfaces).

The Guidelines package is less expensive than the OASYS version and therefore may be more attractive if all you need is MS-DOS compatibility. Note that the Guidelines package is definitely designed for professional developers. It does very little hand-holding for inexperienced programmers. Installation of Guidelines' C++ is easy and straightforward, and invoking the translator is well explained, but from there you are on your own.

With both the OASYS and the Guidelines C++ you need a C compiler. If you already have the Microsoft C compiler, you are all set. Otherwise, you'll have to buy one before you can use these two packages.

A new kid on the block is Zortech C++. Zortech's C++ is a unique implementation in that it is not an AT&T ported translator, but rather a native code compiler. In other words, it reads C++ source files and writes MS-DOS object files. Zortech's is not only a C++ compiler, but it also supports full ANSI C. Zortech's C/C++ compiler is written by Walter Bright of Datalight C fame. In fact, Zortech has purchased full marketing rights to Datalight C, improved it, and added C++ support.

I haven't had a chance to use Zortech's C++. I received one of the first copies just as this article was due. I've read the first few chapters of the manual (about 600 pages), and it looks excellent. This is definitely a hand-holder. Besides that, the package includes a fully integrated development environment (like Turbo C or Quick C) and a pop-up, resident, context-sensitive help facility. And guess what? It's only $99! I will be reporting on my use of Zortech's C++ in the next article.

The Guidelines and Zortech packages will come into play as we deal with MS-DOS topics in the future. Using both the Guidelines and the Zortech packages on the same code will be a good test for the accuracy of the Zortech product since Guidelines C++ is an original AT&T C++ port.

**Designer C++ for OS/2**
$495.00
OASYS
230 2nd Avenue
Waltham, MA 02154
(617) 890-7889

**Guidelines C++ for MS-DOS**
$295.00
Guidelines Software, Inc.
P.O. Box 749, Dept. PJR
Orinda CA 94563
(415) 254-9183 or 9393

**Zortech C++ Compiler**
$99.00
Zortech, Inc.
366 Massachusetts Avenue
Arlington, MA 02174
(617) 643-7969

or in the U.K.:
Zortech Ltd.
106-108 Powis Street
London SE18 6LU, England
01-316-7777

**On with the Code!**

A first example to explain the features of C++ will be a simple object interface. Because almost every program uses the current system date and time, and because that is a clear, simple example of how to define a class, our first class is **DateAndTime**. This will also allow us space in which to concentrate on C++ rather than OS/2, for now. Objects of the type **DateAndTime** supply the programmer with a date and time string in various forms. Listing 1 is the file **datetime.hxx**.

Although Stroustrup calls C++ specific header files **.h** files as in C, OASYS calls them **.hxx** files. The **xx** in **.hxx** stands for ++ (plus plus). OASYS C++ source files end with a **.cxx** extension. Although this is different from Stroustrup's nomenclature, I like the fact that the file name clearly distinguishes between C and C++ sources. (Guidelines and Zortech call sources **.cpp** files and headers, Guidelines has **.h** files, and Zortech **.hpp**—although any name will do for headers.)

The OS/2 system facility for retrieving the date and time is **DosGetDateTime()**. This call uses a data structure to communicate the results; the name being DATETIME. We use the **DateAndTime** class to hide (abstract) the details of this interface although any file that includes **datetime.hxx** needs to include **os2.h**, the OS/2 standard header file. Now let's look at the details of C++ classes and our particular example, the **DateAndTime** class.

The first thing you may notice is that C++ uses a different syntax to mark comments. Each new comment begins with a // (two forward slashes). Older C style comments,

```
/* comment */
```

*Continued . . .*


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

may also be used. In particular instances you may want to use one rather than the other—refer to Stroustrup.

The C++ keyword used to define a class is **class**. A class is very similar to a C data structure. The first block of statements between the words **class**, **DateAndTime**, and **public** are the private members of the **DateAndTime** class. Here, we implement data abstraction. The only way for the user of a **DateAndTime** object to access a private member of the class is through a public member.

The public members of the class are below the keyword **public**. One of the major differences between C++ classes and C data structures is the ability to define class functions—routines that work only for the class and its derived classes. A class function may be either public or private depending on why the class requires it.

The first class function, named **DateAndTime()**, is called the *class constructor*. The constructor is automatically called when you declare an object of type **DateAndTime**. Not every class needs a constructor. A constructor must be provided only when dynamic memory allocation or some other special configuration is necessary.

The second function is called **~DateAndTime()**. This is the *destructor*, or the routine that reverses the effects of the constructor. The destructor is called when an object of type **DateAndTime** *goes out of scope* (is no longer needed). Like the constructor, the object destructor is called automatically (the call is implicit).

In the case of **DateAndTime**, its constructor allocates some memory and automatically retrieves the date and time. If you declare an object of type **DateAndTime**:

```
aFunction()
{
    DateAndTime Now;
    // ...
}
```

the object **Now** already contains the current date and time information. The scope of the object **Now** is within **aFunction()**. When **aFunction()** returns to its caller, **Now** goes out of scope, and the destructor is called automatically.

If you are concerned that the object **Now** will go "out of date," don't be. You can easily make sure that the date and time are current by calling the next class function, **Update()**. You call **Update** this way:

```
aFunction()
{
    DateAndTime Now;
    // ...
    Now.Update();
    // ...
}
```

At first glance, you may think that using data abstraction would cause a lot of function call overhead. Every time you want to update the private member **dtBuf**, you must call **Update()**, which in turn calls the OS/2 routine **DosGetDateTime()**! But before you quit reading, take a closer look.

**Update()** is actually an *inline* expanded function. When a routine is not only declared but also defined (fully written) in a class definition, it is called an inline function. This means that the line inside the **Update()** curly braces ({ }) is directly generated in place of a function call to **Update()**:

```
// where
Now.Update();
// this is generated
DosGetDateTime( (PDATETIME) Now.dtBuf );
```

This concept is very similar to using a **#define**, but using an inline controls the scope of the function's use. Also, C++ fully handles local variables and the like inside an inline function.

In general, when to use an inline function and when not to (save memory vs. save call time) is not a hard and fast rule. However, it is clear that the use of inline functions with data abstraction is necessary in order to minimize overhead.

The next three functions are used to retrieve strings representing the date and time in various forms. The first, **Show()**, returns a pointer to a 26 byte character string in the form,

```
Fri Jun 17 16:35:04 1988\n\0
```

This is in the style of the standard C library **ctime()** routine.

The next two functions, **ShowDate()** and **ShowTime()**, are used to retrieve the date-only and time-only, respectively. Note at this point that these two facilities are made optional. Look back to the **DateAndTime** constructor. It optionally takes two arguments, **dt** and **tm**. If one or both of these arguments are not supplied, they are defaulted to 0. That is another feature of C++—being able to specify default values of arguments not supplied by the caller. So, if you wanted a date-only option but not a time-only option, you could declare **Now** this way:

Get To Know Your Programs Inside! and Out!
Now you can analyze your programs with unprecedented detail with Inside!, a new software package from Paradigm Systems.
Inside! allows you to examine the route your programs take through execution counts, minimum, maximum and total elapsed times and a count of how many times each source line executes—function by function—for popular Borland and Microsoft languages!
Now available
Inside! Turbo C
Inside! Turbo Pascal
Inside! Quick C
Inside! Quick Basic
Inside! Microsoft Pascal
Inside! Microsoft Fortran
Inside! Lattice C
Inside! Logitech Modula-2
PARADIGM
SYSTEMS-INCORPORATED
Paradigm Systems Incorporated
$75.00 Each
P.O. Box 152 Milford, MA 01757
To Place Orders (800)537-5043
Product Support (508)478-0499
Visa/Mastercard Accepted
Inside! is a trademark of Paradigm Systems Incorporated.
Number 195 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
// explicit dt of 1; implicit 0 for tm
DateAndTime Now(1);

// or this
// explicit 0 for tm
DateAndTime Now(1, 0);
```

If you want the best of both worlds:

```
// use both date-only
// and time-only
DateAndTime Now(1, 1);
```

will do. If you attempt to use **ShowDate()** or **ShowTime()** without having initialized them, the string **Error\n** is returned.

The last three routines, **Size()**, **DateSize()**, and **TimeSize()**, return the lengths of the strings that **Show()**, **ShowDate()**, and **ShowTime()** return, respectively. These inline routines are provided in case you want to derive a class from **DateAndTime**, or if you want to allocate memory to store the strings.

Next, look at Listing 2. The **datetime.cxx** file contains the routines that are not inline class functions. Each of these functions identifies itself as belonging to the class **DateAndTime**. Each has the first name **DateAndTime** followed by two colons (**::**), so the constructor for this class is declared as

```
DateAndTime::DateAndTime(int dt =0, int tm =0)
{
    // ...
}
```

Notice as you look through Listing 2 (**datetime.cxx**) how the constructor and destructor never have a type declaration (as **DateAndTime::Show()** does—type **char ***).

Also, as you look through Listing 2 notice how the members of the class are used directly by name. Notice the constructor in this case:

```
dtBuf = new DATETIME;
```

In C, if a function takes a pointer to a data structure as an argument, it must de-reference each structure member using a construct like

```
xyz_type_pointer->dtBuf = new DATETIME;
```

C++ knows that it is working on the member of the object for which the routine was called. The C++ translator does some programming for you. C++ knows when you say,

```
Now.Show();
```

that it needs to translate the statement to something like

```
DateAndTime::Show( &Now );
```

Therefore, a pointer to the object of a given class for which the member function is invoked is implicitly passed as the first argument to the member function. Since the argument is implied, it is not declared as an argument to be received by a function. The pointer can be used with the name **this**, so the same statement above could have been written,

```
this->dtBuf = new DATETIME;
```

This leads to another curious C++ operator, **new**. The **new** operator is used to allocate memory dynamically in C++. This operator might be seen as a mixture between standard C's **malloc()** and **calloc()**, but it's easier to use. For instance, the statement,

```
dtBuf = new DATETIME;
```

allocates memory large enough to hold a DATETIME and returns a pointer to the memory in **dtBuf**. Arrays of a given type can be allocated this way:

```
showBuf = new char[ showSize=26 ];
```

Since the expression is evaluated in the same way that expressions in C are, the statement above could be read as

```
this->showSize = 26;
this->showBuf = new char[ this->showSize ];
```

where the value within the square brackets is the amount of objects of type **char** that the array should contain.

Did you notice the declaration for the private members **monthName** and **dayName**? They are declared static within the class. That means that all objects of type **DateAndTime** will reuse the same month and day name arrays. A new array does not have to be allocated for each object!

The remainder of the constructor—the call to **Update()**—is self-explanatory. And it is likewise easy to see that within the destructor, the C++ **delete** operator does the opposite of **new**. It gives memory back to the free storage manager.

The final three functions, **Show()**, **ShowDate()**, and **ShowTime()**, are used to format the given date and/or time string and return it to the requestor. You could display the returned string as follows:

```
DateAndTime Now;
```

*Continued . . .*


**C Math Toolbox©**
Volume I
**89**
**essential math functions**
**$89.00**

A productivity tool for modern C programmers

**Specifically designed for use with the Microsoft QuickC® and C Optimizing Compiler® 5.x under MS DOS®**

Statistics
Complex numbers
Matrices, and more

For more information, contact your local dealer, or order by mail from:

SILVER STATE SOFTWARE
1000 E. William Street, Suite 100
Carson City, NV 89701
**800/624-0894 ext 874**
**Visa and MasterCard Welcome**

QuickC, C Optimizing Compiler, and MS DOS are registered trademarks of the Microsoft Corp.

Number 186 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

```
printf("%s", Now.Show());
```

Later you'll see the C++ way of writing to the standard output device (the terminal).

### Derived Classes

As was mentioned above, deriving one class from another allows the programmer to add new features to an existing class, without risking damage to the existing class.

If you think for just a minute, you can imagine several features that could be added to the **DateAndTime** class. For instance, you may like to have a way to conveniently access the individual components of the date and the time (not by parsing the returned string!). Besides, there are components that the standard **Show** routines don't even consider, such as the timezone and hundredths of seconds. You may even want an easy way to reset the system date and time from the same object. To do some of these things, we will derive a new class from the **DateAndTime** class.

Look back to Listing 1 (**datetime.hxx**). Notice the second data structure, **DateAndTimePlus**. This new data type is derived from the original **DateAndTime** class. It provides a way to access the individual components of the **dtBuf** structure. A few details must be explained so that you can clearly understand the syntax.

First, a derived class *itself* automatically has full access to the public members of the *base* class. However, to make the public members of the base class available to the user of the derived class, as if a *derived* were really a *base*, the base is made public:

```
struct DateAndTimePlus : public DateAndTime
```

So now, when you declare an object of type **DateAndTimePlus**, it can be treated just as a **DateAndTime** would be:

```
DateAndTimePlus Now(1, 1);
printf("%s", Now.ShowTime());
```

Second, if the derived class is to have access to the private members of its base class, it must be a *friend* of the base. This syntax is displayed in the first line of the **DateAndTime** class. It is interesting to note that the base class chooses its friends, instead of the derived class's choosing which base it will like. You can see that this is a reasonable protection for the base.

The constructor and the destructor for the derived class do little more than help the base to get set up. Now, besides being able to display the date and time, the derived class allows the programmer to peek at the private components of its base class's **dtBuf**.

To illustrate a detail of C++ structures and classes, the derived class was purposely defined using the **struct** keyword rather than the **class** keyword. This is because the only time there is an advantage to using **class** rather than **struct** is when you are hiding details, such as data, from the user. The **DateAndTimePlus** data type has no additional information for hiding. It merely implements an extended interface for retrieving additional information from a **DateAndTime**:

```
DateAndTimePlus Now;
printf("%s", Now.Show());
int h = Now.GetHundredths();
```

But what of the idea to reset the system date and time? Given the OS/2 interface for setting,

```
DosSetDateTime((PDATETIME) dtBuf);
```

(where the cast to PDATETIME is a far (32-bit) pointer to a structure of type DATETIME), what would you do with or without deriving another class?

### Testing the Interface

Finally, look at Listing 3, which shows how I tested both the **DateAndTime** and the **DateAndTimePlus** classes. Figure 1 is an example of its typical output. Here you are introduced to the Stroustrup method of handling file streams and the standard streams, **cin** (standard input), **cout** (character output—shown here) and **cerr** (standard error).

```
Then: Thu Jun 30 13:28:53 1988
Now : Thu Jun 30 13:28:54 1988
Date: Thu Jun 30 1988
Time: 13:28:54
Hour: 13
Min : 28
Sec : 54
Hund: 15
Day : 30
Mon : 6
Year: 1988
Zone: -1
WDay: 4
```

**Figure 1 - Output from DT.EXE**

As you can see, the results of the various class functions are sent to **cout**. The way that this is done looks a bit strange at first:

```
cout << "Hello there, what is your name: ";
char inBuf[64];
cin >> inBuf;
```

Basically, output streams (class ostream) flow this way <<, and input streams (class istream) flow that way >>.

This introduces another C++ concept, operator overloading. As you will see in the next article, it is possible to redefine the meaning of given C (and C++) operators to accommodate more "natural" use of them within a given class. In other words, given the class xyz, you can tell C++ how you will treat the *assignment* (=), the *addition* (+), the *subtraction* (–), etc., of *xyzs*.

### What's Next?

The next time around, we will apply object oriented principles to the OS/2 Interprocess Communications (IPC) facilities. Then we will definitely spend much more time on OS/2. The IPC interface will be much more demanding of C++ than **DateAndTime** was.

I am extremely interested in reading your constructive ideas, code, and arguments for or against C++. Send your responses to Aspen Scientific, Dept C++, PJ, P.O. Box 72, Wheat Ridge, CO 80034-0072. Perhaps this will be the beginning of a great forum. ♦

*Vaughn Vernon is president of Aspen Scientific, producers of CURSES and FORMATION, Unix user interface cross development tools for MS-DOS and OS/2. Vaughn is co-author of the* Advanced C Programmer's Guide to OS/2 *published by Microsoft Press.*

*Code follows.*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ CODE

All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386. Mail order information on page 89.

## Code from "Object Oriented OS/2" by Vaughn Vernon

1

**Listing 1**

```
// datetime.hxx
//
// defines the OS/2 date and time classes
//
// (c) Copyright 1988 Aspen Scientific
// All Rights Reserved.

// the date-and-time class definition

class DateAndTime {

friend struct DateAndTimePlus;                  // allow use of private
members

        // the following is for private class use

        DATETIME        *dtBuf;                 // OS/2 date and time buffer
        char            *showBuf;               // format buffer
        char            *showDateOnly;          // format buffer for date
        char            *showTimeOnly;          // format buffer for time
        static char     *errorMsg;              // an "error" message
        static char     **monthName;            // month names
        static char     **dayName;              // names of days of week
        int             showSize;               // the size of the show buffer
        int             dateSize;               // the size of the date only
        int             timeSize;               // the size of the time only

        // the following is for public use
public:

        DateAndTime(int dt =0, int tm =0);      // constructor
        ~DateAndTime();                         // destructor

        // make date and time current
        void            Update()        { DosGetDateTime((PDATETIME) dtBuf); }

        char            *Show();                // return string for full info
        char            *ShowDate();            // return only a date string
        char            *ShowTime();            // return only a time string

        // return the size of the date and time buffers
        int             Size()          { return (showSize); }
        int             DateSize()      { return (dateSize); }
        int             TimeSize()      { return (timeSize); }

};

// now, derive a class from DateAndTime to handle
// looking at each member of the dtBuf structure.

struct DateAndTimePlus : public DateAndTime {

        // we have no data to hide, only public functions to declare.
        // therefore, we will use a struct, rather than a class.

        // the constructor: it takes the same arguments as
        // its base, and calls the base constructor with the
        // arguments.  the constructor for this class actually
        // does nothing more than help to construct its base.

        DateAndTimePlus(int dt =0, int tm =0) : (dt, tm) { }

        // the destructor: see the constructor

        ~DateAndTimePlus() { }

        int     GetHours()      { return ( dtBuf->hours ); }
        int     GetMinutes()    { return ( dtBuf->minutes ); }
        int     GetSeconds()    { return ( dtBuf->seconds ); }
        int     GetHundredths() { return ( dtBuf->hundredths ); }
        int     GetDay()        { return ( dtBuf->day ); }
        int     GetMonth()      { return ( dtBuf->month ); }
        int     GetYear()       { return ( dtBuf->year ); }
        int     GetTimeZone()   { return ( dtBuf->timezone ); }
        int     GetWeekDay()    { return ( dtBuf->weekday ); }

};
```

2

**Listing 2**

```
// datetime.cxx
//
// example of using classes in C++ for the OS/2 date and time interface.
//
// (c) Copyright 1988 Aspen Scientific
// All Rights Reserved.

#include        <stdio.h>
#include        <os2.h>
#include        "datetime.hxx"

static char *months[] = {
        "Jan", "Feb", "Mar",
        "Apr", "May", "Jun",
        "Jul", "Aug", "Sep",
        "Oct", "Nov", "Dec"
};

static char *days[] = {
        "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
};

// the following are the class object routines

// the constructor of a DateAndTime object
DateAndTime::DateAndTime(int dt, int tm)
{
        // allocate memory for the date and time
        dtBuf = new DATETIME;

        // allocate memory for the show buffers
        showBuf       = new char[ showSize=26 ];

        showDateOnly = (dt ? new char[ dateSize=17 ] : (dateSize=0, (char
          *)0));

        showTimeOnly = (tm ? new char[ timeSize=10 ] : (timeSize=0, (char
          *)0));
```

*Code continues . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Vernon Code, Continued

3

```
        // it is a comfort to know that all objects of type
        // DateAndTime will reuse error, month and day data, since
        // the two class members are declared as static.

        // error message
        errorMsg = "Error\n";

        // set up month and day names
        monthName = months;
        dayName   = days;

        // initialize with the current date and time.
        // perform last, so that the result is accurate.
        Update();
}

// the destructor of a DateAndTime object
DateAndTime::~DateAndTime()
{
        // give memory back to free storage
        delete dtBuf;
        delete showBuf;
        if (dateSize)
                delete showDateOnly;
        if (timeSize)
                delete showTimeOnly;
}

// show the DateAndTime; format a string and return a pointer to it.
// format is the same as that found in the std-C library routine ctime().
char *DateAndTime::Show()
{
        sprintf( showBuf, "%s %s %02d %02d:%02d:%02d %4d\n",
                dayName[ dtBuf->weekday ],
                monthName[ dtBuf->month - 1 ],
                dtBuf->day,
                dtBuf->hours,
                dtBuf->minutes,
                dtBuf->seconds,
                dtBuf->year );

        return ( showBuf );
}

// show only the date portion of the DateAndTime object
char *DateAndTime::ShowDate()
{
        if (showDateOnly == (char *)0)
                return ( errorMsg );

        sprintf( showDateOnly, "%s %s %02d %4d\n",
                dayName[ dtBuf->weekday ],
                monthName[ dtBuf->month - 1 ],
                dtBuf->day,
                dtBuf->year );

        return ( showDateOnly );
}

// show only the time portion of the DateAndTime object
char *DateAndTime::ShowTime()
{
```

4

```
        if (showTimeOnly == (char *)0)
                return ( errorMsg );

        sprintf( showTimeOnly, "%02d:%02d:%02d\n",
                dtBuf->hours,
                dtBuf->minutes,
                dtBuf->seconds );


        return ( showTimeOnly );

}
```

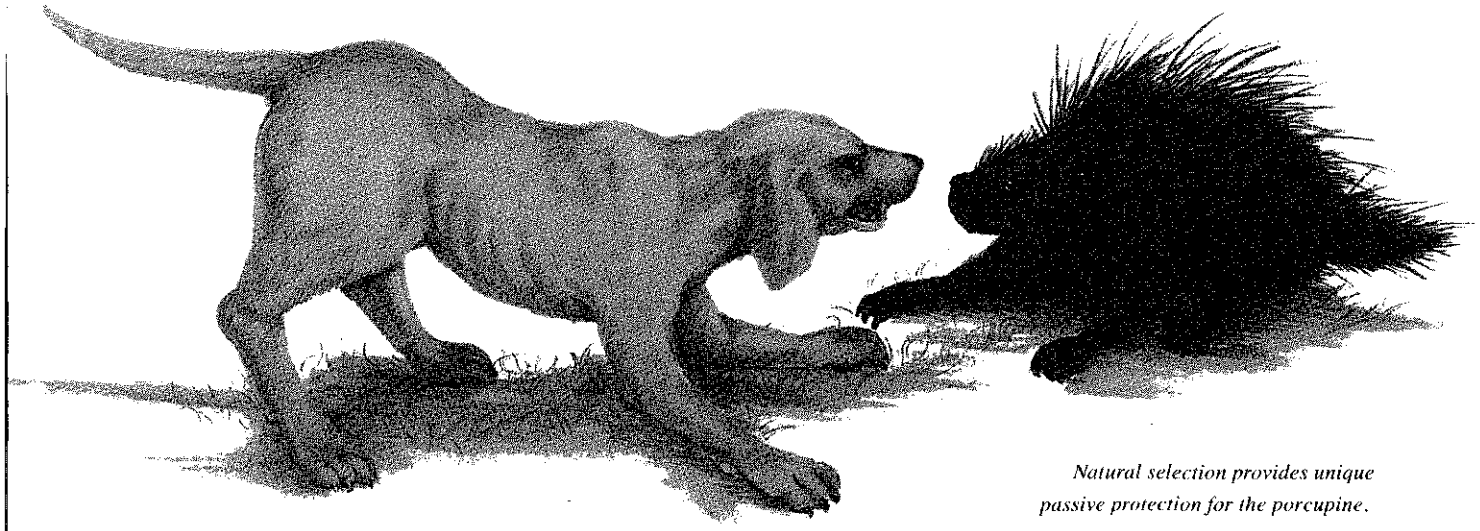
**Listing 3**

```
// test program for the OS/2 date and time classes
//
// compile/link with OASYS: ccxx dt.cxx datetime.cxx

#define INCL_DOSPROCESS         // directs os2 header definitions
#include        <os2.h>
#include        <stream.hxx>
#include        "datetime.hxx"

// program start-up

int main()
{
        DateAndTime Then;
        cout << "Then: " << Then.Show();

        // go inactive for 1000 milliseconds, or one second
        DosSleep( 1000L );

        DateAndTimePlus Now(1, 1);
        cout << "Now : " << Now.Show();

        // show the date and time separately
        cout << "Date: " << Now.ShowDate() << "Time: " << Now.ShowTime();

        // now use the newly available routines from the
        // derived class DateAndTimePlus

        cout << "Hour: " << Now.GetHours()          << "\n";
        cout << "Min : " << Now.GetMinutes()        << "\n";
        cout << "Sec : " << Now.GetSeconds()        << "\n";
        cout << "Hund: " << Now.GetHundredths()     << "\n";
        cout << "Day : " << Now.GetDay()            << "\n";
        cout << "Mon : " << Now.GetMonth()          << "\n";
        cout << "Year: " << Now.GetYear()           << "\n";
        cout << "Zone: " << Now.GetTimeZone()       << "\n";
        cout << "WDay: " << Now.GetWeekDay()        << "\n";

        exit( 0 );
}
```

◆



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED



*Natural selection provides unique passive protection for the porcupine.*

# The Block -- Natural Selection For Software Protection

*Inventor and entrepreneur Dick Erett explains how "The Activator" provides sane protection for your intellectual property.*

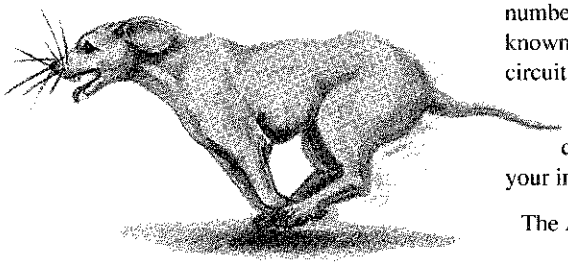
"In any industry, just as in nature, the process of natural selection raises one solution above another. Natural selection is the most elegant of engineers.

In the area of software protection The Block has been selected by the marketplace as the solution that works. Over 500,000 packages are protected by our device.

For the past 4 years our philosophy has been; *'You have the right and obligation to protect your intellectual property.'*

## A New Ethic For Software Protection

In allowing end-users unlimited copies of a software package and uninhibited hard disk and LAN operation, The Block has created a new ethic for software protection.



By removing protection from the magnetic media we remove the constraints that have plagued legitimate users.

They simply attach our key to the parallel port and forget it. It is totally transparent, but the software will not run without it.

## A New Technology For Software Protection

Our newest model, The Activator, builds on our current patented design, and establishes an unprecedented class of software protection.

We have migrated and enhanced the circuitry of The Block to an ASIC (Application-Specific Integrated Circuit) imbedded in The Activator.

This greatly improves speed and performance, while reducing overall size. Data protection can also be provided.

## Programmable Option

The Activator allows the software **developer** the option to program serial numbers, versions, or other pertinent data known only to the developer, into the circuit, and access it from the program.

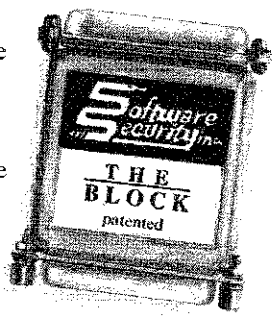
Once you program your part of the chip, even we have no way to access your information.

The ASIC makes emulation of the device virtually impossible. It also presents an astronomical number of access combinations.

## Full 100% Disclosure

Since The Activator is protected by our patent we fully disclose how it works. Once you understand it, endless methods of protection become evident.

Just as no two snowflakes are the same, no two implementations of The Activator are identical. And like the snowflake the simplicity of The Activator is its greatest beauty.




We never cramp your programming style or ingenuity. Make it as simple or complicated as you desire.

Let us help safeguard what's rightfully yours. Please call today for additional information or a demo unit. *It's only natural to protect your software.*"

**1-800-333-0407**
**In Connecticut 203-329-8870**
**Fax 203-329-7428**

Software Security inc.

**870 High Ridge Road**
**Stamford, CT 06905**

Unlimited Copies • Programmable • Small Size • Fast • Patented • Data Protection

The Block and The Activator are trademarks of Software Security

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# Ada—
# A Software Engineering Tool

by
**Richard D. Riehle**

*Robert Frost once said that writing free verse was like playing tennis with the net down. The game under those conditions requires great self-discipline. In computer programming we often play with the net down. Ada puts up the net, adds several referees, and installs electric-eye sensors on the fault line and boundaries.*

As a response to a "software crisis," the U.S. Department of Defense (DoD) decreed in 1979, that all new software development should be performed using Ada. Then the DoD created a waiver process so that nearly anyone with a good imagination could create a rationale for avoiding the transition to Ada. The result: Ada didn't have enough launch-pad thrust to reach orbit. Another factor in Ada's slow acceptance was a reluctant IBM. Without the IBM endorsement, any new language has trouble gaining acceptance. According to *Aviation Week and Technology*, IBM finally put its full weight behind Ada after its Federal Systems Division lost several million dollars in government contracts that required Ada.

Now Ada's popularity is increasing in velocity: internationally in Europe and Japan; academically in the university software engineering community. The National Aeronautics and Space Administration (NASA) has adopted Ada. The Federal Aviation Administration has chosen Ada for the new air traffic control system. The University of Santa Clara now requires Ada instruction in its electrical engineering curriculum. Ada is even gathering a following in the commercial and MIS marketplace. CRI, Inc., of Santa Clara, California, has developed a relational database product in Ada for use in Ada systems.

**Estimate of Ada Market Potential in U.S. Aerospace Industry**

| 1986 | 1987 | 1988 | 1989 | 1990 | 1991 | 1992 |
|---|---|---|---|---|---|---|
| Thousands of People Developing Software | | | | | | |
| 100 | 112 | 125 | 140 | 157 | 176 | 197 |
| Percent Working on Ada Programs | | | | | | |
| 3% | 6% | 12% | 20% | 30% | 40% | 50% |
| Thousands of People Developing Software in Ada | | | | | | |
| 3.0 | 6.7 | 15.0 | 28.0 | 47.0 | 70.4 | 98.5 |
| Investment per Person Working on Ada Programs *(in Thousands)* | | | | | | |
| $10 | $10 | $10 | $10 | $10 | $10 | $10 |
| Ada Support Marketing Opportunity *(in Millions)* | | | | | | |
| $30 | $67 | $150 | $280 | $471 | $704 | $985 |

*Assumption: Number of software developers grows 12% per year*
*Source: Rational, Mountain View, California*

In April 1987, Undersecretary of the Army James Ambrose declared an end to waivers and mandated that all Army projects be developed in Ada, including MIS applications. Since then all branches of the DoD have elevated their commitment to Ada, and most of the defense contractors have gotten the message. Ada is now alive and well—in fact, thriving—and is a viable language alternative for any software development project.

Why do we need another programming language? And why Ada? And what is an "Ada," anyway? Then again, what is all this nonsense about "software engineering"? Is that just another fancy term for programming in the way that "sanitary engineer" is another name for janitor? In this series of articles, we hope to answer these and other related questions about Ada. We'll discuss the premises on which the language was designed and its differences from other languages, and we'll examine the issue of fulfilled and unfulfilled expectations.

### Language Characterisitics

Ada is a descendant of Algol-68 via Pascal. If you know Pascal, Algol, or PL/1, you will find much that's familiar in Ada. However, Ada is a very formal language, and many words and phrases take on specific, new meanings when describing Ada concepts. Also, Ada adds new capabilities to its ancestral languages and moves in the direction of "object-oriented design" (OOD). But Ada is not a "pure" OOD language in the image of Smalltalk, Objective C, or ACTOR.

Before we proceed with more detail, let's take a look at some of the major characteristics of the language. First a definition. We use the word *type* a lot in Ada. A type defines both the permitted set of values and the legal operations for an object. Objects may be discrete data items (scalars), composite data items, or entire executable modules.

Some of Ada's more prominent features are:

- multiple levels of abstraction
- strong typing
- strict scope and visibility rules
- high modularity
- object-oriented design
- built-in exception handling
- built-in concurrent/real-time processing capability
- separation of specification code from implementation code
- separate compilation of modules
- incorporation of current concepts of software engineering

One of Ada's unique features, *generic* components, enables Ada programmers to build context-independent modules.



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

### The Software Engineering Imperative

Without an awareness of Ada's software engineering foundations, it will be hard to understand why the code sometimes reads like the software equivalent of a Bach fugue.

During the past twenty-five years intensive scholarly research into the programming process has resulted in concepts like "formal proof of correctness" (Dijkstra), Structured Analysis and Design (DeMarco), problem/solution space models (Ledgard), and "information hiding" and "levels of abstraction" (Parnas). There have been hundreds of other contributors to this evolving discipline, and a visit to a university bookstore will now turn up plenty of titles that include the expression "software engineering."

### Goals of Software Engineering

The designers of Ada adopted the goals of software engineering defined by Ross, Goodenough, and Irvine:

- Modifiability
- Efficiency
- Reliability
- Understandability

These software engineering goals are no different from the implied goals in our day-to-day programming, but now they are explicitly stated—almost codified.

An additional goal for Ada is "portability": There is only one Ada. No dialects of the language are permitted. Source code must compile in any Ada environment. The Ada Joint Programming Office (AJPO) of the DoD validates every Ada compiler. An unvalidated compiler isn't Ada. This is the first computer language subjected to such a rigorous standard. Any attempt to corrupt Ada will be rebuffed. She's ". . . just not that kind of girl."

### Principles

The goals of software engineering led to a definition of underlying principles. By principles, we do not mean "methods." A method is something like "structured analysis." A principle is the foundation for the method.

Since software engineering is an emerging discipline, there is no complete agreement on all of the principles, but the principles most commonly associated with Ada are also defined in the work of Ross, Goodenough, and Irvine:

- Abstraction
- Information hiding
- Modularity
- Localization
- Uniformity
- Completeness
- Confirmability

Notice the absence in this list of Warnier-Orr, Jackson System Development, software metrics, object-oriented design. These are "methods," which would be based on the principles.

### Abstraction

One of the most important principles in software engineering is "levels of abstraction." By abstraction we mean: Show only the essential properties of a program without revealing the details. There is a presumption that we can decompose an abstraction into its components.

Ada is expressly designed to enable multiple levels of abstraction. We find this principle represented in Ada by packages, generic program units, distinctions between unit specification and unit body, among abstract data types, and between private and limited-private types, and by the ability to define entirely new types.

An example software abstraction familiar to many designers is the Data Flow Diagram (DFD). The "context diagram" represents the highest level of abstraction of the DFD. Subsequent DFD levels describe subordinate levels of abstraction until we are at the most elementary (non-decomposable) level.

### Information Hiding

Information hiding is closely related to abstraction. We simplify the use of program units by hiding unnecessary information. Information, here, is defined in a very broad sense and includes details about algorithm implementation, data types, and objects.

Information hiding is not a new principle. It has been available to us in one form or another from the earliest days of programming. One example is the OPEN command found in many languages. We issue the

```
OPEN (parm1, parm2)
```

statement and are spared the tedious effort of coding our own device driver interface, exception handling, etc. Ada, by its

### Who Is Ada?

Ada is often called "the first computer programmer."

Augusta Ada Byron, daughter of Lord Byron, was born December 8, 1815, in England. She was always addressed by her middle name Ada. When her father left to carouse about the continent, Ada's mother Lady Byron set about educating Ada in mathematics as a moral discipline. Ada continued to study mathematics throughout her life.

Ada became friends with Charles Babbage, inventor of the Difference Engine and the Analytical Engine. An Italian engineer, Luigi Menebrea, wrote a short paper describing the Analytical Engine, and one of Babbage's friends suggested that Ada translate the paper into English.

When Ada undertook the task of translating Menebrea's paper, she also decided to make a few "notes." The notes, labeled A thru G, were three times the length of the original paper. It is in these notes that Ada inadvertantly immortalized herself in computer history.

The "notes" describe programming methods far beyond what was then possible with the Analytical Engine. Ada's mind took a leap into the future. One hundred years before Eniac, she was describing looping constructs, branching, subroutines, variables, and GIGO (garbage in, garbage out). Ada even touched lightly on Artificial Intelligence. Of course, Ada did not use our contemporary computer argot, but the notes read almost like one of the popular books of our day that describe the possibilities of electronic computers.

When in 1979, the U.S. Department of Defense High Order Language Working Group (HOLWG) finally decided to accept the language design of Jean Ichbiah's group at France's Honeywell/Bull, the language needed a name. According to Grady Booch, author of *Software Engineering with Ada*, "Jack Cooper of the Navy Material Command evolved the perfect name for this new language: Ada . . ."

It seems appropriate that an organization as committed to feminist issues as the Department of Defense should name its new programming language in honor of a nineteenth century woman.—*RDR* •



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

very design, expands upon and actually enforces this principle. In later discussions we'll examine data types that provide elegant ways to hide implementation details.

### Modularity

Ada encourages program design using small, well-defined modules. Ada programmers have standard methods for separate compilation, data and procedure encapsulation, the creation of "loosely-coupled" objects, and top-down structural design.

The phrase "loosely-coupled" is an important concept in Ada's rendition of object-oriented design. The loosely-coupled object in Ada is a *generic* unit that can be used over and over in many different programs independently of the specific contexts of those programs. In fact, building generic "reusable components" is a key feature of Ada.

An example of loose-coupling in the real world is in your automobile. You can't take the carburetor from your Chevy and put it on my Ford. The carburetor and the engine are tightly-coupled objects. On the other hand, your twelve-volt battery is a loosely-coupled object that I can steal from your Chevy and use in my Ford.

This leads us to the concept of "software ICs." Many integrated circuits are loosely-coupled. Often, an electrical engineer may select generic ICs (objects) from a catalog to create a unique hardware design. Object-oriented design strives to develop reusable software units, or components, as generic as those integrated circuits!

### Localization

Localization is almost the inverse of modularity. But localization stresses the cohesiveness of the objects in a module. This tends to make individual modules small, logically concise, and easy to modify. We see here that much of Ada's design relates to the modifiability goal. Small, loosely-coupled, highly-cohesive modules are easier to modify and maintain. They are also easier to create.

Ada permits an encapsulated *type* that can be affected only by the operations defined within the scope of a package. This is achieved via *private* and *limited*-private data types. The end result is "abstract data types."

### Uniformity

We could also use the word consistency. Every programming organization has its standards. Some use indentation on the line after the { while others insist on data name normalization and still others are totally laissez faire. Often, "correct" coding style is subje ct to argument. The principle of uniformity is related to the goals of modifiability and understandability. The idea, of course, is to keep the style the same.

Ada supports uniformity by virtue of its rigorous structure. Ada programmers build software systems as small, well-defined modules (Ada *packages*). The implementation coding style of a particular package may be peculiar, but the package user never sees the algorithmic code. All access to the package is through the package specification, and that substantially limits the variations of style available to the package developer. Even here Ada doesn't let us play with the net down.

### Completeness

How can we know our solution is complete? No programming language alone can make this happen, but there are those Ada enthusiasts who insist that Ada's structure and environment can help.

### Confirmability

By confirmability, we mean some method of determining that our program is correct. Edsgar Dijkstra's "formal proof of correctness" for software would be ideal here. Unfortunately, neither Ada nor any other language advances us to that level of confirmability.

Ada's modularity enables us to separately compile and test program components. In addition, strong data typing provides tools to enforce confirmability. As a stongly "typed" language, Ada encourages design that includes defining new, rigorously constrained data types. And Ada's built-in exception handling permits us to implement our own error and exception routines.

### Software Engineering Beyond Ada

No language can provide, by itself, all the tools necessary for fulfilling the goals and principles of software engineering. But Ada is more than a language. Compiler developers are also supplying the Ada Programming Support Environment (APSE). In future articles we'll explore different implementations of APSE.

Other elements of software engineering are also necessary—project management tools, structured methods (analysis, design, walkthroughs), prototyping, quality assurance, Computer Assisted Software Engineering (CASE) tools, and software metrics.

### The Ada Language

Now that we have reviewed some of Ada's underlying goals and principles, we can look at the language itself. In keeping with the principles of software engineering, we start at the highest level of abstraction, the *package*.

The package is unique to Ada. We can find concepts in other languages that roughly correspond to the Ada package but nothing that is as complete. By a package we mean a collection of logically related objects. This collection can be data, data types, related subprograms, and type declarations. A package consists of two parts: the *package specification* and the *package body*.

The user of a package usually has no need to see the details of the implementation (package body). The only part of a package the programmer would see is the package specification. The specification is the programmer's window into the package.

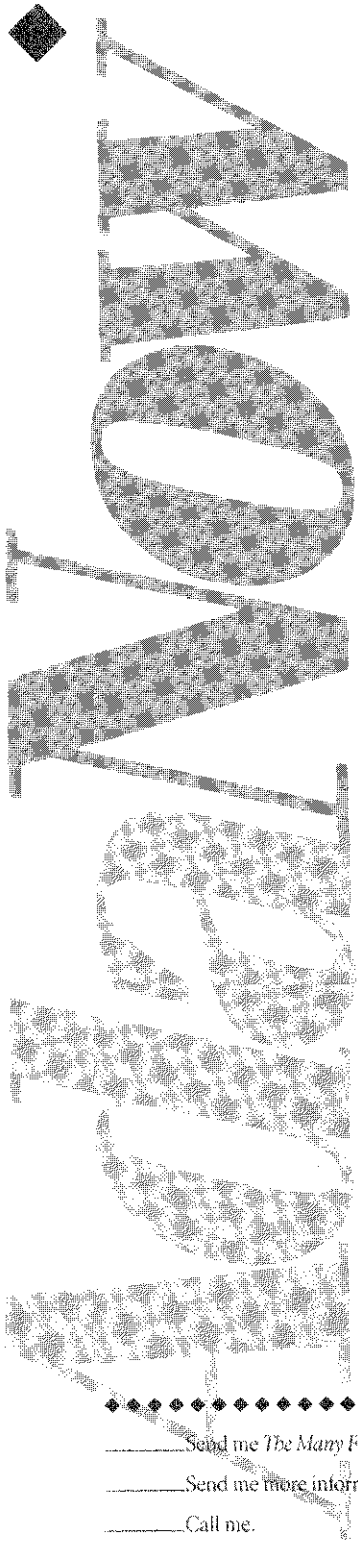
There are software companies that specialize in creating both generic and non-generic Ada packages. This is a grow-

**Ada Reserved Words**

| | | | | |
|---|---|---|---|---|
| abort | declare | generic | of | select |
| abs | delay | goto | or | separate |
| accept | delta | | others | subtype |
| access | digits | if | out | |
| all | do | in | | task |
| and | | is | package | terminate |
| array | else | | pragma | then |
| at | elsif | | private | type |
| | end | limited | procedure | |
| | entry | loop | | |
| begin | exception | | raise | use |
| body | exit | mod | range | |
| | | | record | when |
| | | | rem | while |
| | | new | renames | with |
| case | for | not | return | |
| constant | function | null | reverse | xor |


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Introducing PC AT Version 3




# The Serious Ada Compiler For Serious Ada Programmers

If you're a serious Ada programmer today, or expect to be one—if you're serious about DoD business and those new DoD directives—you look for a lot more in your Ada compiler than just validation. And the new fourth-generation Alsys Ada compiler Version 3 for the PC AT and compatibles gives it to you.

Take code quality, for example. Version 3 generates the highest quality code of any Ada compiler on any machine! Check the PIWG benchmarks, and those of U-Michigan. Compare the quality with code from the mature C and Pascal compilers you're used to. Check especially where you might need quality most, and where Version 3 shines—in the implementation of procedure calls and exceptions. The elimination of code associated with unused subprograms gives you large reductions in code size in many applications.

Consider the Runtime Executive. True production quality. No exception-handling overhead is incurred unless an exception is actually raised. The Runtime is optimized for programs running in limited memory, or running for a long time. That's serious.

Consider the Ada-specific high level optimizer technology (not derived from Fortran or C) that removes redundant constraints checks and does so much more for code quality.

Consider robustness and reliability. Version 3 is written in Ada and bootstrapped through itself. That's proof right there that it will compile 400,000 lines of Ada code. Beyond validation, we test our compilers on hundreds of thousands of lines of extra code—from the ACEC tests, and from Ada Repository programs, and from our own specially designed code that breaks most other compilers.

There's a lot more that's serious about Version 3. The new Multi-Library environment, for example, that maximizes the efficiency of programming teams; error messages that correct, instruct, and speed the programming process; human engineering; superb documentation and customer support.

Send for our free brochure *The Many Facets of Quality.*

alsys

In the US: Alsys Inc., 1432 Main St., Waltham, MA 02154 Tel: (617) 890-0030
In the UK: Alsys Ltd., Partridge House, Newtown Rd., Henley-on-Thames, Oxon RG9 1EN Tel: 44 (491) 579090
In the rest of the world: Alsys SA, 29 Avenue de Versailles, 78170 La Celle St. Cloud, France Tel: 33 (1) 3918.12.44

_____ Send me *The Many Facets of Quality.*

_____ Send me more information on the PC AT Version 3 Compiler.

_____ Call me.

Name_____

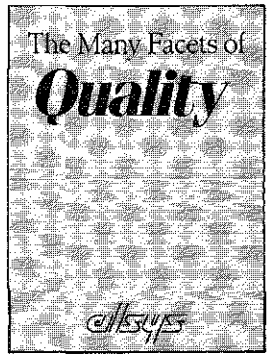
Company_____

Address_____

City_____ State_____ Zip_____

Phone_____

Alsys, Inc. • 1432 Main Street • Waltham, MA 02154




Number 118 on the Reader Service Card

'Ada is a registered trademark of the U.S. Government (AJPO).


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Why Do We Need a New Language?

In his book on Software Engineering, Henry Ledgard describes what sets the professional apart from the amateur programmer: The completed work of a professional must be reliably usable and maintainable by someone other than the author. Moreover, the professional typically develops software that will interact with a large body of other software written by other professionals.

There are other issues. Professional programmers and programming managers know that the vast majority of time in any software organization is devoted to maintenance of existing code. In the Department of Defense it is estimated that 80% of the software dollar goes into maintenance. Not only is this a thankless task, but it also propagates new bugs. Additionally, because there is no "science" of programming, each coder does things differently. Attempts at discipline over the years have been dismally unsuccessful. Imagine maintaining the code for all the systems of the DoD (or any government agency) and transporting the code into a worldwide military command and control system or two hundred logistics systems.

There is also the old issue of re-inventing the wheel. Most of the code we create does the same task as some other code written by someone else. Comparatively little original software is coded. The ideal is to make code as re-usable, generic, and simple as paperclips. This is a non-trivial problem in a small organization like your hometown bank; certainly non-trivial in an organization the size of the DoD.

Now imagine the cost of all this duplication, error-prone maintenance, and managerial oversight. It's enough to make any self-respecting general want to give up nuclear weapons and return to clubs and stones.

Ada is designed for professional programming/software engineering. The typical Ada project is very large, consists of many diverse components, and must be faultlessly reliable. The system is usually constructed by a team of programmers working on independent modules, and each module (package) must work properly with all the other modules. Professor Phil Schrodt of Northwestern University has stated a rule of thumb: "No good program can be written by more than ten people. The best programs are written by one or two people." Ada presumes to repeal Professor Schrodt's rule of thumb.

It became clear in the late 1960s that the DoD was in a "software crisis." By the early 1970's the DoD was using over 400 different programming languages. Maintenance of the systems programmed with these languages was becoming a nightmare, and the very computer software that was being implemented for the defense of the country was becoming part of the threat.

When a government agency has a problem, the first thing it does is establish a committee. In this case the committee, formed in 1975, was named the High Order Language Working Group (HOLWG) and consisted of both defense and civilian members. HOLWG developed criteria for the "ideal" DoD language and surveyed the languages then available. HOLWG decided that none of the languages was consistent with DoD requirements.

In 1977, HOLWG issued a request for proposal for a new computer language. It received 15 proposals and selected four finalists labeled Red, Green, Blue, and Yellow. In the next round of refinements, the Red and Green languages were selected. Finally, in 1979, the Green language, submitted by a team at Cii-Honeywell/Bull of France, was selected. The team was headed by Dr. Jean D. Ichbiah.

Ada was designed according to current concepts of software engineering. Some say it is the first language to be developed that way. It is, however, not without its critics. Many software engineers believe that a better language that implements the principles is Modula-2, developed by the author of Pascal Niklaus Wirth. Some say that Ada is the last great language of the 1960s and point to the emergence of 4GL (Fourth Generation Language) technology. Still others dislike the "size" of the language, comparing it to the current trend towards Reduced Instruction Set Computers (RISCs) instead of Complex Instruction Set Computers (CISCs).

The jury is still out on the eventual fate of Ada. At present it seems to be healthy and gaining strength. Just a few years ago, many people had their doubts. According to a bulletin released by the Ada Joint Programming Office (AJPO) in December 1987, there are now over 120 compilers available for the language. Moreover, organizations that have adopted Ada and implement its "reusability" features report dramatic improvements in programming productivity.

There is hardly a mainframe or minicomputer environment for which there is no compiler. As you might expect, there are outstanding Ada compilers for the DEC VAX series. Strangely, though, the system best designed to take advantage of Ada's tasking feature, the Cray series, does not yet have a "validated" compiler. By "validated" we mean that the compiler has successfully compiled and executed the 1,850 programs contained in the AJPO test suite. And the rule is, until it is validated by the AJPO it isn't Ada. Recent estimates are that there will be about 150 validated Ada compilers by Summer of 1988.

There are some good implementations of Ada for the IBM/Clone PC environment. For a low-cost Ada compiler, try RR Software, Inc.'s JANUS/Ada (C-PAK priced around $100). Recently, Meridian Software introduced a pre-validated version of Ada for the Macintosh. Meridian and Alsys also have validated compilers for MS-DOS environments.—*RDR* ♦

### Ada Compilers for Microcomputers

This list of Ada compilers is not comprehensive, nor should its entries be construed as specific recommendations.

| | | Validated | Environment |
|---|---|---|---|
| Janus/Ada | RR Software, Madison, WI | Yes | PC/Clone |
| AdaVantage | Meridian Software, Laguna Hills, CA | Yes | PC/Clone, |
| | | Soon | MacIntosh (under MAC OS) |
| | | Yes | Z-8000 |
| Alsys | Alsys, Inc. Waltham, MA | Yes | PC/Clone, |
| | | Yes | Macintosh (under AUX) |
| | | | Motorola 68000 family |
| | | Yes | PS/2 |
| | | Soon | OS/2 |
| Ada-86 | Softech, Inc. Waltham, MA | Yes | Cross-compiler from Vax series targeted at the INTEL 8xx86 series |
| NYU Ada/Ed | New York University | | |

NYU Ada/Ed is a low-cost "Ada" interpreter designed for learning something about Ada. However, you can't do much with it and it's not at all suitable for production programming. Ada/Ed will execute most of the programs in the book by Clark (see the bibliography), but watch out for integer sizes.


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

ing business, but there is still a need (and a marketplace) for better packages.

Here is a skeleton package that incorporates other packages. Note that Ada comments are preceded by two hyphens ( -- ):

```
with OPERATIONS_RESEARCH;              -- begin package specification
package BOX_CAR_SCHEDULER is
    type STATUS_TYPE  is (FULL, EMPTY);     -- defined  data type
    type SERIAL_NO  is new INTEGER;         -- derived  data type
    BOX_CAR_STATUS : STATUS_TYPE;           -- assign a data type
    BOX_CAR        : SERIAL_NO;             -- assign a data type
    function  BOX_CAR_IS(BOX_CAR :in SERIAL_NO) return
     STATUS_TYPE;
    procedure PLACE_BOX_CAR(PARM1;PARM2;PARM3...);
            .
            .
end BOX_CAR_SCHEDULER;                 -- end package specification

package body  BOX_CAR_SCHEDULER        -- begin implementation
        .                                      -- algorithms
        .                                      -- are in this part
end BOX_CAR_SCHEDULER;                 -- end implementation
```

In this example we are creating a package to solve the classic problem of moving empty and full box cars in a railroad network. The package will use another package, OPERATIONS_RESEARCH, which is incorporated into this package by the Ada *with* statement; then we define two data types, STATUS_TYPE and SERIAL_NO, and create two variables and assign a type to each; then we specify the functions and procedures that will be found in the body of the package. The most important thing to observe here is that the package specification may be the only thing available to anyone who wants to use the package BOX_CAR_SCHEDULER. The package body may be hidden. With this specification, another programmer can create a new program unit (package, procedure, or function). For example:

```
with TRACK_MANAGEMENT;
with PERSONNEL;
with ENGINE_MANAGEMENT;
with BOX_CAR_SCHEDULER;
package RAILROAD_MANAGEMENT is
        .
        .
end RAILROAD_MANAGEMENT;           -- end package specification
```

The package body of BOX_CAR-SCHEDULER and other packages are hidden from the programmer writing the package, RAILROAD_MANAGEMENT. And the package body for OPERATIONS_RESEARCH was hidden from the creator of BOX_CAR_SCHEDULER.

### Subprograms

Ada has two kinds of subprograms: *procedures* and *functions*. The difference between the two is quite simple. A function returns a result as part of an expression. A procedure is a simple statement; for example, a simple procedure to convert non-metric to metric:

```
with TEXT_IO;
procedure METRIC is
   INCHES, FEET, YARDS : INTEGER;
   METERS : FLOAT;
   package INT_IO is new TEXT_IO.INTEGER_IO(INTEGER);
        function METER_CONV(IN,FT,YDS:INTEGER) return FLOAT is
             INCH_TOTAL         : INTEGER;
             NEW_METRIC         : FLOAT;
             CONVERSION_FACTOR : constant FLOAT := 0.02540;
        begin
             INCH_TOTAL := (YDS * 36) + (FT * 12) + IN;
             NEW_METRIC := FLOAT(INCH_TOTAL) * CON-
                VERSION_FACTOR;
                   -- Notice the conversion of INTEGER type to
                   -- FLOAT type using FLOAT(INCH_TOTAL)
               return NEW_METRIC;
        end METER_CONV;
begin
        INT_IO.GET(YARDS);
        INT_IO.GET(FEET);
        INT_IO.GET(INCHES);
        METERS :=  METER_CONV(INCHES,FEET,YARDS);
end METRIC;
```


# Software Tools
For Programmers & Non-Programmers

★ NEW ★ VERSION **Opt-Tech Sort**™ ★ NEW ★ VERSION

Opt-Tech Sort is a high performance Sort/Merge/Select utility. It can read, sort and write a file faster than most programs can read the data. Example: 1,000 records of 80 bytes can be read sorted and a new file written in less than 9 seconds (IBM XT). Opt-Tech Sort can be used as a stand-alone program or Called as a subroutine to over 30 programming languages.

Numerous features are included. A partial list of features includes: The ability to process files of any size. Many filetypes are supported including Sequential, Random, delimited, Btrieve, dBASEII & III and many others. Up to 30 key fields can be specified over 16 different types of data are supported. Powerful record selection capability allows you to specify which records are to be included on your output. Record reformatting allows you to change the structure of your output record and to output special fields such as record numbers for use as indexes. **Newest Features: Faster than ever! LIM EMS support. User exits to pass or receive records. Less disk space required and much more!** MS-DOS $149.

## On-Line Help™

On-Line Help allows you to easily add Help Windows to all your programs. On-Line Help is actually two help packages in one. You get BOTH Resident (pop-up) and Callable help systems.

The resident version allows you to add help to any system. Your Help system is activated when the "Hot Keys" that you specify are pressed. You can then chain between help windows in any manner you desire. Upon clearing the help windows you are right back into what ever program you left.

The callable version allows you to easily display help windows from your programs. A simple Call to the help system makes the window appear. On-Line Help is callable from over 20 different languages.

You have full control over the help window content, size, color and location. MS-DOS $149. Demo $10.

## Scroll & Recall™

Scroll & Recall is a resident screen and keyboard enhancement. It allows you to conveniently scroll back through data that has gone off the top of your display screen. Up to 27 screens of data can be recalled or written to a disk file (great for documenting systems operations). Also allows you to easily recall and edit previously entered DOS commands without retyping.

Scroll & Recall is very easy to use. Its a resident utility that's always there when you need it. MS-DOS $69.

Visa, M/C, AMEX, Check, Money Order, COD or Purchase Orders accepted

To order or to receive additional information just call and receive immediate highly qualified attention!

**Opt-Tech Data Processing**
P.O. Box 678 / Zephyr Cove, NV 89448
(702) 588-3737

Number 16 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Here we have a function call within a procedure. In addition, we have a procedure call, GET, from INT_IO.

Illustrations like this often elicit a ho-hum response from experienced programmers. There's nothing here that couldn't be done in some other language. The directive **with** is somewhat equivalent to C's **#include** or Pascal's **{$I ...}**, but there are some advantages in clarity. For example, the value to be returned in an Ada function is always explicitly returned. Also, we do an explicit type conversion on INCH_TOTAL in the expression that computes CONVERSION.

The first thing we see in this procedure is the invocation of another package, TEXT_IO. Ada doesn't have its own input/output procedures, so input and output are controlled by Ada packages. Several packages are provided as part of the standard language implementation: SEQUENTIAL_IO, DIRECT_IO, TEXT_IO, and LOW_LEVEL_IO. Other _IO packages are available from Ada software vendors and the AJPO. These include console, screen, and window handling packages, graphics packages, and packages for device drivers.

Since we want to enter integer numbers via a keyboard, we need to use the package TEXT_IO, which contains three generic packages named INTEGER_IO, FLOAT_IO, and FIXED_IO.

An Ada generic does not exist as an executable entity. It is often referred to as a "template." To use any generic object (package, procedure, or function), you must supply the characteristics of the objects to be processed and create a newly named version of the generic object. This is called *instantiation*, meaning to create a new "instance" of the generic object. In Ada the parameter(s) for a generic may be a type, a variable, or even another subprogram.

INTEGER_IO, as a generic package within TEXT_IO, allows us to send and receive numbers from/to a device in ASCII format. INTEGER_IO automatically converts the numbers to the proper internal INTEGER format so that we can perform calculations. The generic does not exist as a working package until we instantiate it. In this case we instantiated INTEGER-IO as a new package named INT_IO for data type INTEGER.

This may seem a little redundant, but suppose we had really wanted to constrain the GET for each object in the procedure. We might have created the following new data types:

```
type INCH_TYPE is range 1..12;
type YARD_TYPE is range 1..144;
type FEET_TYPE is range 1..3;
```

then created objects of those types,

```
INCHES : INCH_TYPE;
FEET   : FEET_TYPE;
YARDS  : YARD_TYPE;
```

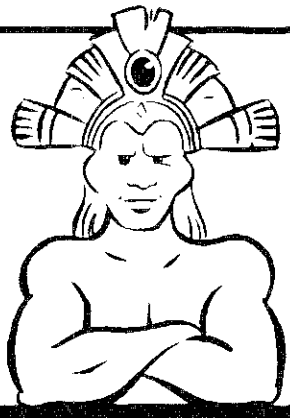
and then instantiated INTEGER_IO for each type

```
package INCH_IO is new TEXT_IO.INTEGER_IO(INCH_TYPE);
package FEET_IO is new TEXT_IO.INTEGER_IO(FEET_TYPE);
package YARD_IO is new TEXT_IO.INTEGER_IO(YARD_TYPE);
```

Now our procedural coding would read:

```
INCH_IO.GET(INCHES);
FEET_IO.GET(FEET);
YARD_IO.GET(YARDS);
```


EVERLOCK



COPY PROTECTION

Designed for user-transparency, clone compatibility & strength. It features:

- no need for damaged media or I/O plugs
- supports all Hard & Floppy disk formats
- file-server networks supported
- variable number of installs (0-99)
- create demos with remote unlock option
- allows protected upgrades by modem BBS
- limit program use by date or count

$195 Starter Kit or $495 with NO meter counts. Free info & demo disk available

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 (816) 776-2700

Number 51 on the Reader Service Card

*For those who need a*

**FAST C COMPILER** with 8087 & software floating point and a full DOS 3 compatible STDIO library
**FAST 8086 ASSEMBLER** that is stand-alone, or inline with no separate assembly step
**FAST LINKER** that makes stand-alone .EXE files with both memory and disk-based overlays
**SOURCE DEBUGGER** for both C and Assembler
**FULL SCREEN EDITOR** that edits 2 files with no size limits & can invoke the Compiler & Linker
**GRAPHICS** package with line and text functions

DeSmet DC88

We've combined our popular DeSmet C88 Development Package with the DeSmet D88 Debugger and our Graphics Package. Library source code is available separately.

$99

C Ware
P.O. Box 428, Paso Robles, CA 93447
Phone:(805)239-4620 Telex: 358185 BBS: (805)239-4834
We accept VISA, MC & AMEX. Call now and we'll ship today
945 Spring St #14, Paso Robles, CA 93446

Number 38 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

If the newly instantiated package, INCH_IO, receives input outside the range of 1 through 12, a constraint error will be raised at runtime. This is one reason Ada code sometimes seems a little baroque.

Some relief is available in the form of the *use* option. Rather than explicitly name TEXT_IO as the parent of INTEGER_IO, we could have said,

```
with TEXT_IO; use TEXT_IO;
        .
        .
package INT_IO is new INTEGER_IO(INTEGER);
use INT_IO;
        .
        .
GET(INCHES);
```

Many Ada programming shops prohibit or discourage the **use** option because of the need for absolute clarity and control over the visibility and scope of every element of the program. This may be of little consequence in a small software product but becomes a major issue in a large software system designed to control parallel processing in a "mission critical" environment.

## Tasks

One unique feature of Ada is the *task*, a program unit devoted to concurrent processing. The task may be used in either a multiprogramming or multiprocessing environment. The task has no direct analogue in other languages. Historically, we resort to assembler language or some special purpose language we access via CALL constructs to match Ada's task.

The task becomes an especially important programming construct as supercomputing and parallel processing emerge. With Ada we can design a system in which we launch twenty (not a limit) parallel tasks, all of which can communicate with each other. Tasks can be used with great efficiency in complex simulation programs where it is often convenient to break a problem into small pieces and merge results at different stages of completion.

The overall structure of a task is similar to that of a package:

```
task  PACEMAKER is
        .                   -- Specification for a task
        .                   -- in an embedded system that
        .                   -- controls a heart pacemaker
end   PACEMAKER;

task body PACEMAKER is
        .                   -- Algorithmic implementation
        .                   -- is placed here
        .
end  PACEMAKER;
```

An important aspect of Ada is the ease with which we can create programs for "embedded systems," by which we mean software systems consisting of multiple programs and/or processors that operate independently of any human interference. Examples would be radar guidance systems, missile telemetry, automated medical monitoring and control devices, unmanned space vehicles, etc. Reliability is the most important attribute of these kinds of systems.

Ada's charter to be the language of embedded systems is one reason for the rigorous discipline it enforces on the software engineer.

We can create multiple tasks that rendezvous with each other in a variety of ways. There are methods for prioritizing, starting, stopping, and monitoring tasks. One task may start several others and wait until one of those others completes


Celebrating our 10th Anniversary!

AMX™ 86

KADAK's engineers bring years of practical real-time experience to over 600 installations world-wide.

This *real time*
MULTITASKING KERNEL
simplifies *real life*
product development

- No royalties
- IBM PC DOS® support
- C, PL/M, Pascal
- Preemptive scheduler
- Time slicing if needed
- Essential source code included
- Intertask messages
- Dynamic operations
  - task create/delete
  - task priorities
  - memory allocation
- Event Manager
- Semaphore Manager

***AMX86™ v2.0 for 8086/88, 80186/88, 80286 systems***

| | |
|---|---|
| Demo package | $25 US |
| *Manual only* | $75 US |
| AMX86 system | $2195 US |
| Full source Add | $805 US |

(Shipping/handling extra)

*Also available for 8080, Z80, 68000*

KADAK Products Ltd.
206 - 1847 W. Broadway
Vancouver, B.C. Canada V6J 1Y5
Telex: 04-55670
Fax: (604) 734-8114
Telephone: (604) 734-2796

Number 211 on the Reader Service Card

DISK FORMAT CONVERSION

XenoCopy-PC

$79.95 + $5.00 S/H
Sales Tax if CA.

PC-DOS program lets your PC

VISA MasterCard UPS COD

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from all previous versions
for only $25.00 Call for Authorization

To Order Contact:

XENOSOFT™

2210 Sixth Street, Berkeley, CA 94710
(415) 644-9366

Number 123 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

some assignment. The important thing about tasking is that we can have more than one task operating concurrently.

In a later article we'll explore more of the fascinating world of Ada tasking. There are those who believe that until you understand tasking, you don't understand Ada. The potential for Ada's task in such non-defense uses as reliable embedded software in health care and commercial robotics appeals to many software people. In the long run, Ada's use in non-military environments will be more important than its military uses.

### Exceptions

Another unique feature of Ada is its built in *exception* handling capability. There are exception conditions that Ada will "raise" due to constraint errors, program errors, numeric errors, storage errors, and tasking errors. You can also create your own exception conditions in a variety of ways and raise your own exceptions. The syntax for exception handling is

```
begin
  -- implementation of some program statements
exception
     when PROGRAM_ERROR =>
  -- perform some program statements to
  -- handle the exception
end;
```

The emphasis on strong exception handling, like that on tasking, is due to Ada's role in creating embedded systems. We can't have a space vehicle suspend its mission due to a constraint error and orbit around Pluto until the programmer arrives to fix the problem. Very few programmers want to work in the neighborhood of Pluto.

### Summary

In future articles we'll examine specific examples of Ada's code. These examples will include data processing, mathematical methods, tasks, and systems programming techniques. We'll also look at how Ada deals with bit manipulation and low-level input/output associated with device-driver development.

Keep in mind that Ada was not designed for small, one-person projects. Ada's charter is to make large, complex software development efforts more manageable; and, since 60 to 80 percent of our software dollars go into the maintenance of existing code, to improve the efficiency of program modification. The real savings from Ada won't be realized until the next generation of programmers has to tinker with the code we are creating now. And this is an issue that MIS managers should be exploring with a sense of urgency. The DoD is not the only organization experiencing a "software crisis."

### References


DeMarco, T. *Structured Analysis and System Specification.* New York: Yourdon Press, 1978.

Dijkstra, Edsger W. *A Discipline of Computer Programming.* Englewood Cliffs, N.J.: Prentice Hall, 1976.

"IBM Forges Links to Ada Vendors to Enhance Role in Aerospace Market." *Aviation Week & Space Technology* (March 28, 1988): 67–69.

Jackson, M.A. *Principles of Program Design.* New York: Academic Press, 1975.

Ledgard, H.L. and M. Marcotty. *The Programming Language Landscape.* Chicago: Science Research Associates, 1981.



TurboC, Lattice C, and QuickC

"c_wndw" Windows, Menus & File Access

*Now there is an affordable, complete set of functions to produce end-user DOS applications in the C language. "c_wndw" is a very complete toolkit, providing the following facilities*

- o Color and cursor control
- o Pull-down and pop-up menus, and dialog lines
- o Field formatting and string handling
- o Entry editing of numbers and dates
- o Stacked, overlapping windows
- o File access error trapping and handling
- o Access to dBASE III plus '.DBF' files
- o Hashed random file access
- o Function key mapping and control
- o Undo and Help capability

*For a license fee of $35, we provide a full set of object libraries for a C compiler, the full manual, and 19 sample programs. (S&H $2, GA res. add $1.85 tax). Specify your C compiler. Source code is available for $95.*

To find out more about the "c_wndw" toolkit, write or call for information to Marietta Systems - (404) 565-1560
P.O. Box 71506, Marietta, GA 30007 Visa-M/C

*Trademarks: "c_wndw", dBASE, QuickC, Turbo C, Lattice C*

Number 223 on the Reader Service Card

SPECIAL REPORT

COMPUTER VIRUS PROTECTION!

***The threat of computer viruses is real!***

**FILE INTEGRITY CHECK**ing system performs a thorough check of all of the programs and files on your hard disk(s). Its complete reporting methods can alert you to the potential presence of a virus in your system. Works on all IBM PC, XT, AT, PS/2 and true compatibles running DOS.

**Our checking methods include:**

* Subdirectory names * Added/Deleted/Renamed Files/Dirs
* Filenames * Attributes * Time * Date * Size * CRC * MCRC
* Thorough, complete reports

Our unique MCRC (Modified CRC) check was developed because a virus could possibly alter a file yet keep the same CRC - if that happens, our MCRC check would catch it! Nothing is more effective than our dual checking methods.

**The best part is the price: $15.** Not much to pay for peace of mind. In addition, the FILE INTEGRITY CHECKing system comes with six months of free access to our BBS which deals with the topic of computer viruses 24 hour a day! Send your check today or order by phone with your VISA or M/C.

GILMORE SYSTEMS
213-275-8006
P.O. Box 3831, Beverly Hills, CA 90212-0831

Number 135 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Parnas, D.L. "On the Criteria to Be Used in Decomposing Systems Into Modules." *Communications of the ACM* 15:1053–8.
Parnas, D.L. "A Technique for Software Module Specification with Examples." *Communications of the ACM* 15:330–6.
Ross, D.T., J.B. Goodenough, and C.A. Irvine. "Software Engineering: Process, Principles, and Goals." *Computer* (May 1975).

### Select Bibliography

There are several good books on Ada. At last count, the Ada Joint Programming Office (AJPO) listed about 85 titles. These are some that I have found valuable.

Barnes, J.G.P. *Programming in Ada*. Menlo Park, Cal.: Addison-Wesley, 1984. A short introduction for the programmer with some math background. Well written and chock full of good examples. Excellent treatment of tasking.
Booch, Grady. *Software Engineering with Ada*. Menlo Park, Cal.: Benjamin/Cummings, 1987. This is a classic work on Ada—Mr. Booch is an Ada guru extraordinaire. The book is very thorough and uses a case history method. Good for the serious student with a strong math background, definitely not for the casual reader or the faint-hearted.
Caverty, Phillip and Philip Goldstein. *Introduction to Ada: A Top-down Approach for Programmers*. Monterey, Cal.: Brooks/Cole, 1986. The two Phil's cover a lot of good material in this little book. Excellent examples and not too mathematical. A good book for the data processing programmer.
Clark, Robert G. *Programming in Ada: A First Course*. Cambridge England: Cambridge University Press, 1985. If Ada is the first programming language you are going to study, this is a good book. The examples are simple, the writing clear. The problems are within the scope of a novice programmer. You can use the NYU Ada/Ed interpreter to test most of the examples in the book. Little discussion of Ada's more powerful features such as tasks, generics, limited-private types, access types, or abstract data types, but will satisfy the new programmer.
Gilpin, Geoff. *Ada: An Introduction and Tutorial*. New York: Brady Communications, 1986. One of the best self-study books on Ada I've seen. The writing is excellent. There is a problem with clear labeling of the examples, but Gilpin's treatment of record types, private and limited-private types, and exception handling is superb. If you want to read only one book to get an understanding of Ada, this is the one, but keep Grady Booch handy for reference.
U.S. Department of Defense. *Reference Manual for the Ada Programming Language*. Washington, D.C.: ANSI/MIL-STD_1815A-1983. The one book everyone should have, but don't try to learn Ada from it unless you are a born-again masochist.

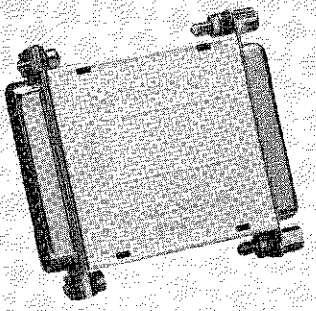
### Two Good Biographies of Lady Augusta Ada Lovelace

Baum, Joan. *The Calculating Passion of Ada Byron*. New York: Archon Books, 1986. Includes generous excerpts from Ada's "Notes."
Stein, Dorothy K. *Lady Lovelace's Notes: Technical Text and Cultural Context*. Cambridge, Mass.: MIT Press, 1985. A less admiring portrait of Ada; skeptical of her actual achievements and/or ability in mathematics.

And if you are serious about Ada, get on the mailing list of the Ada Joint Programming Office, Department of Defense OUSD (R&E), Washington, D.C. 20301. ♦

*Richard D. Riehle is an instructor in Ada at Merritt College in Oakland, and Foothill College in Los Altos, California. He is also an independent computer consultant. Mr. Riehle's training organization ICE (for Intensive Computer Education) conducts seminars on a variety of computer subjects. Mr. Riehle has over twenty years in software development, including design, programming, and project management. He is currently working on a book,* From COBOL to ADA, *a tutorial for experienced business programming professionals. Readers may contact Mr. Riehle at 1090-105 Tanland Drive, Palo Alto, CA 94303, (415) 858-1267.*

Hard Locks for Soft Parts.



At Rainbow Technologies, we think protecting software developers' investments is very serious business. That's why we designed the first fully effective security solution for software running on PCs and other computers.

Our family of virtually impenetrable Software Sentinel hardware keys provides the highest level of software protection the developer can get. While remaining invisible to the end user.

Take a look.

Key Sentinel Family Features.

Prohibits unauthorized use of software ▫ No need for copy protection ▫ Unlimited backup copies ▫ Virtually unbreakable ▫ Pocketsize key ▫ Transparent operation ▫ Transportable

Software Sentinel.
- Runs under DOS and Xenix on IBM PC/XT/AT and compatibles
- Algorithm technique (Never a fixed response.)
- Serial or parallel port version
- Minimal implementation effort
- Higher level language interfaces included
- 100 times faster than hand-held response devices (Dyna)

Software Sentinel-C.
- Higher level language interfaces included
- Runs under DOS on PC/XT/AT and compatibles
- Parallel port version only

Software Sentinel-W.
- Designed for workstations, superminicros and minicomputers
- Serial port only (modem-style)
- Algorithm technique
- We provide detailed interface specifications. Developer creates a port driver.
- Interface is asynchronous, 25-pin DCE/DTE, DB25S, RS232/RS422/RS423
- Easily ported to OEM's CPU from software supplied. OEM's CPU must support serial port.
- Software Sentinel-W is supported on DEC, IBM, Sun, AT&T, Intel, Motorola, Data General, Prime, Harris, Perkin-Elmer, Concurrent, Ridge, Pyramid, Celerity, Encore, Gould, Fortune, Charles River Data, Plexus, Masscomp, Altos, Convergent Technologies, Tandy 16/6000, Wang, Unisys, Tektronix, Cromemco, Zilog, Bull, Olivetti, Siemens, Stratus, Tandem, Everex, Arix, Pacific Micro, Logic Automation, Ferranti, Ferranti-Packard, Ferranti-Cetec, Ferranti-Dowty, Ferranti-Ortenberg, Ferranti-Elbit, Ferranti-Sperry, Ferranti-Computer Systems, Ferranti-Cetec Computer Corporation, Ferranti-Dowty Computer Systems, Ferranti-Elbit Computer Systems, Ferranti-Ortenberg Computer Systems, Ferranti-Sperry Computer Systems, Ferranti-Computer Systems Limited, Ferranti-Cetec Computer Corporation Limited, Ferranti-Dowty Computer Systems Limited, Ferranti-Elbit Computer Systems Limited, Ferranti-Ortenberg Computer Systems Limited, Ferranti-Sperry Computer Systems Limited

# The REXX Language on the PC

by
**Andrew J. Chalk, Ph.D.**

Every programmer who has used batch files extensively knows that the DOS batch facility is rarely dignified with the description "programming language." It limits the programmer so much as to prevent his or her doing some very basic programming tasks. For example, you cannot ask the user a question in a batch file and then (make your program) act on the response. You cannot access system information (such as the time of day) and act on the response. You cannot make a batch file switch from an arbitrary directory, load a program, and switch back to the original directory. There is virtually no "structure" to batch files, and elementary program control structures like DO WHILE loops and SELECT statements are missing.

It is no surprise that a myriad of utilities designed to do just these things have appeared on bulletin boards, or that computer magazines can run articles entitled "The Utility that DOS Forgot." Applications software and development tools for microcomputers have advanced by leaps and bounds since the introduction of the PC, but the basic system utility programming tools included with DOS have remained virtually static.

Mainframe programmers are the people most aware of this situation since it is the rule rather than the exception that mainframe operating systems have systems utility languages. Unix has its shell scripts, MVS has job control language, and TSO has CLISTS. Programmers whose computer baptism was into microcomputers may miss a systems utility language less, much as the person who has always used floppy disks does not miss a hard disk. The fact that few users survive a trip in the other direction suggests that hard disks are useful. Likewise, the fact that mainframe programmers miss their systems utility languages when they move to the microcomputer should give us pause for thought— maybe we ought to see what they mean.

This article is about such a systems utility language— REXX, the language that IBM mainframes use under the VM/SP operating system. REXX has been implemented, in full (with the exception of some VM/CMS-specific features), for the PC by Mansfield Software in *Personal REXX*. Mansfield also produces an editor, KEDIT, that is a PC implementation of the IBM mainframe editor XEDIT. Personal REXX is integrated with KEDIT in a way functionally similar to mainframe REXX's integration with XEDIT. Both editors use powerful macros written in their respective versions of REXX. These, and all other implementations of REXX, are interpreted although this may change.

First, we will describe the REXX language and then look at the PC implementation. We will finish with a discussion of how REXX can be used, including source code for a utility to keep track of computer usage. Needless to say, an article of this length can only skim the language and convey a few basic concepts.

## An Overview of the REXX Language

REXX was developed by M.F. Cowlishaw at IBM's U.K. labs during 1979 through 1982. Early versions were distributed only internally to IBM over the company's VNET internal network. REXX spread throughout IBM as a subculture long before its eventual release in 1983, supplanting IBM's official utility languages EXEC and EXEC2. At one point, Cowlishaw was replying to 350 pieces of electronic mail a day from IBM employees in 20 countries. Cowlishaw set down the final language definition in *The REXX Language: A Practical Approach to Programming* (referenced at the end of this article). REXX took on new significance in 1987, when IBM announced at their Systems Application Architecture (SAA) presentation that REXX will be the standard "scripting language" across their product line from PC to mainframe. The objective is that a REXX program written for one system will run unamended on all others.

REXX is best explained with examples. Here is the simplest possible REXX program:

```
Example 1—The World's Simplest REXX Program.
say "Hi y'all world"
```

All this program does is print to the screen

```
Hi y'all world
```

This illustrates one of the most important design principles behind the REXX language: Make the programming statements descriptive of their functions. The equivalent DOS statement is **echo Hi y'all world**, which is not as intuitive but is a reasonable substitute. Note that single or double quotes may delimit a string literal, but if you want to use one literally inside the string, you must use the other as the delimiter.

Cowlishaw's treatise *The REXX Language* begins with a section on fundamental language concepts. The first principle he chooses to discuss is readability. The concept pervades his book—Cowlishaw wants a language that can be used to get things done quickly, and he sees as part of that the ability for the user to read the source code and quickly translate the language concepts into their human language equivalents.

REXX is not only expressive of its purpose in its statements, but it also allows variable names to be descriptive as well. Variables can be almost any length, so if you want to name a variable "encyclopedia," it is quite legal to do so. Let us build on the above example to introduce variables:

```
Example 2—Creating Variables in REXX
say 'Enter a number'
pull number
say 'The number is' number
```

The screen output is

```
Enter a number
5
```



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

The number is 5

There are several new things here. In line 1, the program displays the statement within quote marks on the screen. This is just like Example 1. The second line illustrates one way of soliciting input in REXX. The statement **pull** pauses program execution for input, in this case from the user, but in other instances, from a facility called the *stack* (to be discussed below). The user enters the digit **5**, and REXX assigns it to the variable **number**. Notice that the variable **number** is created simply by assigning it the value typed in by the user (in this case, 5). There is no need for type declarations or variable declarations in REXX. Just make new variable assignments "on-the- fly," and REXX creates the variables.

The final line of the program looks superficially like just another invocation of the **say** statement. However, there is more to it. The variable **number** has been assigned to **5** in the previous line. REXX replaces the variable with its value and prints out the digit **5**. Notice that the single quote marks in the **say** statement are closed before the word **number**.

Two variations on the last example illustrate another general principle behind REXX. If the third line were changed to

```
say 'The number is number'
```

the program would display the line

The number is number

showing that symbols within single quotes are not interpreted by REXX. Instead, they are passed through as literals to the operating system. In this case the REXX statement **say** displays the line on the screen, but if you had entered the following,

```
'c:\wp'
```

the REXX program would start WordPerfect from the root directory of your "C" disk.

Consider another way of expressing the last line in Example 2. Suppose we leave out the single quotes altogether so that our line becomes

```
say The number is number
```

What happens then? The computer displays the line

THE 5 IS 5

on the screen. This is just the opposite of the previous case. Since no symbols are within single quotes, every symbol is interpreted. It so happens that the default in REXX is uppercase. Therefore, all symbols in the example are converted to uppercase by the REXX interpreter and passed to the **say** instruction for display. All single spaces between symbols are maintained, and (if we had any) multiple spaces are converted to a single space. The last word in the program is converted to uppercase, but REXX has already assigned the digit **5** to NUMBER and hence proceeds with the interpretation before the **say** statement displays it to the screen.

Consider example 3:

```
Example 3—Local Variables and Subroutines
say 'Enter a number'
pull number
say 'The number is' number
say 'Twice the number is' 2*number
say 'The square is' square(number)
say 'What power do you want to calculate?'
pull exponent
say 'The answer is' power(number,exponent)
exit

square : procedure
arg number
  return number**2

power : procedure
arg num,exp
  calc=1
  do exp
    calc = num*calc
  end
return calc
```

This program might proceed as follows:

Enter a number

3

The number is 3

Twice the number is 6

The square is 9

What power do you want to calculate?

4

The answer is 81

The first four lines contain nothing beyond the material in Example 2. However, line 5 introduces the use of functions in REXX. When the expression **square(number)** is reached, the opening parenthesis abutted onto the word **square** signifies a function. The interpreter transfers control to the line **square : procedure** and begins execution there. The **arg** instruction informs the function that there is a single parameter being passed from the calling (main) program, and REXX proceeds to evaluate the function. In this example, the calculation is simple. Square the parameter. The statement **return number**2** returns the result to the main program.

It will be apparent from the output from the last line of code that the REXX variable **number** inside **square** is local to that function. This is achieved through the use of the **procedure** instruction following the label **square:**. **Procedure** makes all variables created within or passed to a function local to that function.

The last step in Example 3 is the function **power**. I'm introducing this function to illustrate REXX program control statements. All repetitive control statements in REXX are subsumed under variations of the **do** construct. In this case the simple repetitive **do** loop is used, but this is only by way of example. The function **power()** returns the result of raising **number** to the given power. It is more syntactically complex than **square**, but no more complex conceptually. The REXX language has as rich a set of program control statements as any other high-level language, including **do [repetitor] . . . end, do [repetitor] until . . . end, do [repetitor] while . . . end, if . . . then, else, select . . . when . . . otherwise**, and **do forever** (the **leave <condition>** instruction causes an exit from an infinite loop when **condition** is true). Arrays are implemented using a stem-leaf syntax; that is, **element.1** and **vararray.i.100** are valid references to elements of a one-dimensional and a two-dimensional array, respectively. Since no explicit variable declaration is required, it is legal to increase the size of arrays just by assigning values to new elements. REXX also supports recursion.

*Continued . . .*



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Programs that deal with user input from the keyboard or from files benefit from the string handling abilities built into REXX. The readability of REXX code extends beyond the literal sense of the term to the language's ability to handle statements from conceptually the same standpoint as the English language. Consider the following example:

```
Example 4—String Handling
say 'Type in a sentence'
pull response
say 'Number of words is' words(response)
say 'The length of the sentence is'
length(response)
say 'Here are the first three words'
say word(response,1)
say word(response,2)
say word(response,3)
say 'here are the fifth through twelfth
characters'
say substr(response,5,8)
```

Here is a sample session using Example 4:

Type in a sentence

I just can't think of anything

Number of words is 6

The length of the sentence is 30

Here are the first three words

I

JUST

CAN'T

Here are the fifth through twelfth characters

ST CAN'T

Here we see that REXX can parse a sentence into its constituent English language words, or even letters, with great ease. These and other features make the language well-suited to text-intensive applications. Note the use of the four built-in functions **words()**, **length()**, **word()**, and **substr()**. **Words(string)** returns the number of words in a string, **length(string)** returns its length, **word(n)** returns the word at position **n** in the string, and **substr(string, start, length)** returns the characters in **string** from position number "start" for "length." These built-in functions do not need to be defined in the program, and like all REXX functions they can be nested. My favorite "one-liner" is

```
say center(strip(reverse(string)),80)
```

which reverses the order of the letters in a string typed on the command line. The order of execution is to reverse the string, then strip the blanks, then center the result on a line of 80 characters.

REXX can synthesize words as well as break them down. The double bar (||) is the concatenation operator, and it is used as shown in Example 5.

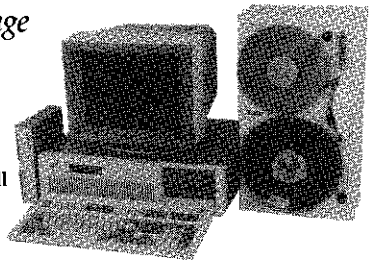
```
Example 5—Using concatenation to build REXX
strings
one = 'mono'
game = 'poly'
train = 'rail'
tree = 'logue'
say one||game one||train one||tree
```

This produces the following string on the screen:

monopoly monorail monologue


# 9-Track Tape Subsystem

*The Data Interchange Solution You've Been Waiting For!*



Qualstar's ½" 9-track Ministreamers™ bring full ANSI data interchange capability to your PC or Macintosh™ computer system.

With 9-track tape, you are free to exchange data files with any mainframe or minicomputer in the world.

Our affordable Ministreamers come in both 7" and 10½" versions and use less desk space than an ordinary sheet of paper. They can provide 1600 thru 6250 BPI capability and may be used for disk backup as well as data interchange. Complete subsystem includes tape drive, coupler card, cables, and software.

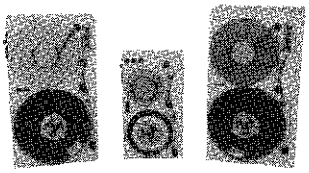
Qualstar has become the number one supplier of desk-top 9-track systems for a good reason; our tape drives have established an outstanding record for reliability and low cost of ownership.

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

**Call us today!**

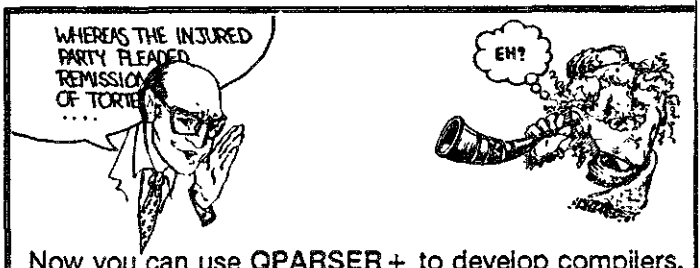
QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822



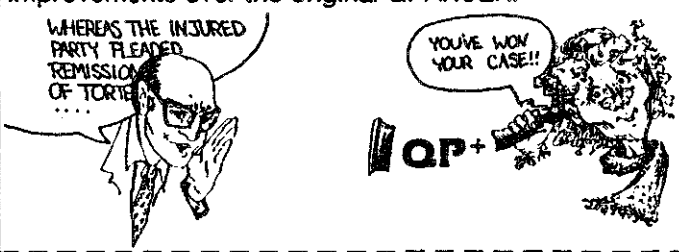
Macintosh is a trademark of Apple Computer, Inc.

Number 97 on the Reader Service Card




Now you can use QPARSER+ to develop compilers, interpreters, complex user-interfaces, language & file format translators (i.e. Pascal to C, Bit Map to Postscript), language debuggers like lint, etc.

Develop language translators in C and Pascal within the IBM PC/XT/AT or VAX/VMS environments. A new user manual, automated syntax tree construction and an advanced code generation language are just a few of the improvements over the original QPARSER.




Another translation by QPARSER+™

Limited Time Offer Just $300 — FREE demo disk available

QCAD Systems
1164 Hyde Avenue, San José CA 95129 (408) 727-6884
Outside Calif. call TOLL-FREE (800) 538-9787

Number 189 on the Reader Service Card




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

In fact, the above case where we had string literals could be written even more transparently as

```
say 'mono''poly' 'mono''rail' 'mono''logue'
```

and string literals can be concatenated with variables, as in,

```
say one'poly' one'rail' one 'logue'
```

which produces the same screen output as Example 5.

In accord with Cowlishaw's desire for a fast development system, a trace facility is an integral component of the REXX language definition. Trace instructions are turned on and off with the REXX TRACE command, and a variety of options are available to control the types of statements traced and set the interpreter into interactive mode so that control pauses between traced statements.

The REXX language itself is high-level (as the examples above make clear) and syntactically is close to other procedural languages such as Pascal. REXX does have built-in functions to handle bit-level operations (BITAND(), BITOR(), AND BITXOR()) and similar functions to convert between character, ASCII ordinal, and hex representations of bytes. The ability of the language to treat variables as strings but also interpret numeric strings as numbers is extended to real numbers as well as integers. The default precision is 9, but can be altered at will.

**Personal REXX for the IBM PC/XT/AT**

On IBM mainframes, the interface between REXX and the system is virtually seamless. In particular, several CMS commands and REXX work closely together. Regular REXX users will wonder whether the PC implementation of REXX is as seamless as mainframe REXX. In this respect, Mansfield has done a creditable job. One CMS feature is a global variable facility, which is a more powerful analog to the DOS environment **set** command. Since this would not otherwise be available in Personal REXX, Mansfield has implemented it as a separate program that you can invoke within REXX. Likewise with the CMS LISTFILE, EXECIO, MAKEBUF, DROPBUF, and DESBUF commands. All of these commands can be called from within REXX and function in a manner similar to their CMS counterparts'.

Personal REXX has 10 PC hardware information functions that return such things as the current video mode, the equipment flag, RAM size, and ROM date. In addition, there are 10 functions to change features of the PC hardware. These include delays, sound effects, direct keyboard read without echo, access to ports, and RAM access by memory address. The DOS-specific functions in Personal REXX provide REXX versions of DOS commands that return a return code to the calling REXX program. Some of these have extensions to the functions available from the basic DOS command. There are also screen handling functions making the development of color menus straightforward. One feature I would like to see in a future release is REXX functions to make interrupt calls and DOS function calls. Further, although Mansfield has gone to some lengths to provide access to hardware functions and the operating system, Personal REXX does not have the system access found in, for example, C or implementations of Pascal such as Turbo Pascal. REXX includes no hardware-specific functions since it is designed to be portable.

You can choose the degree of integration you desire. The tradeoff is that each component of Personal REXX uses


ADD TO THE POWER OF YOUR PROGRAMS WHILE YOU SAVE TIME AND MONEY!

CBTREE does it all! Your best value in a B+tree source!

**Save programming time and effort.**
You can develop exciting file access programs quickly and easily because CBTREE provides a simple but powerful program interface to all B+tree operations. Every aspect of CBTREE is covered thoroughly in the 80 page Users Manual with complete examples. Sample programs are provided on disk.

**Gain flexibility in designing your applications.**
CBTREE lets you use multiple keys, variable key lengths, concatenated keys, and any data record size and record length. You can customize the B+tree parameters using utilities provided.

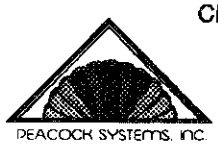
**Your programs will be using the most efficient searching techniques.**
CBTREE provides the fastest keyed file access performance, with multiple indexes in a single file and crash recovery utilities. CBTREE is a full function implementation of the industry standard B+tree access method and is proven in applications since 1984.

**Access any record or group of records by:**
- Get first
- Get previous
- Get less than
- Get greater than
- Get sequential block
- Get all *partial* matches
- Insert key and record
- Delete key and record
- Change record location
- Get last
- Get next
- Get less than or equal
- Get greater than or equal
- Get partial key match
- Get all keys and locations
- Insert key
- Delete key

**Increase your implementation productivity.**
CBTREE is over 8,000 lines of tightly written, commented C source code. The driver module is only 20K and links into your programs.

**Port your applications to other machine environments.**
The C source code that you receive can be compiled on all popular C compilers for the IBM PC and also under Unix, Xenix, and AmigaDos! No royalties on your applications that use CBTREE. CBTREE supports multi-user and network applications.

CBTREE IS TROUBLE-FREE, BUT IF YOU NEED HELP WE PROVIDE FREE PHONE SUPPORT.
ONE CALL GETS YOU THE ANSWER TO ANY QUESTION!
CBTREE compares favorably with other software selling at 2,3 and 4 times our price.
**Sold on unconditional money-back guarantee.**
YOU PAY ONLY $159 - A MONEY-SAVING PRICE!
TO ORDER OR FOR ADDITIONAL INFORMATION
CALL 1-800-346-8038 or (703) 847-1743
OR WRITE




NOW! Variable length records.

Peacock Systems, Inc., 2108-C Gallows Road, Vienna, VA 22180

Number 216 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

memory. At a minimum, a resident interrupt handler occupying 2.5K must be resident in memory at all times. The main REXX program occupies 100K but must also have a storage area allocated for variables it creates. The size of this area is set by the user but must be at least 10K to run a REXX program of reasonable size and can extend up to 40K. To run applications under REXX, Mansfield recommends a minimum of 384K, but someone willing to accept less functionality could use the program on a 256K machine. If memory is available, the REXX interpreter can be made resident. In its base configuration, a REXX program is invoked from the DOS prompt with the command **REXX <programname>**. Most users will also load the REXX batch manager, a resident 2.5K file, enabling them to invoke REXX programs in the same way that batch files are invoked (that is, **<programname>**).

One of the most useful features of REXX is the program stack. This permits you to store instructions for later execution and has no counterpart in the DOS batch facility. To use the stack in Personal REXX requires another resident utility. This one occupies 3K plus the reserved stack area (typically user set at 8K) and may be defined as a device in the CONFIG.SYS.

To summarize what we have so far, full-featured REXX occupies 17.5K to 70.5K (subject to user preferences) while it is not running. Add 100K to for the interpreter and 10K to 40K for an (required) internal storage area when REXX is running. In practice, I decided to run REXX from a ramdisk in extended memory, rather than have it permanently resident, and to load the stack manager (13K) and set the internal storage area at 40K. This occupies a total of 21K when I'm not running a REXX program, and 156.5K when I'm running REXX.

In operation I found Personal REXX to be reliable and easy to use. I used the program with SIDEKICK, CED, and a resident clock all simultaneously in memory. This setup functioned fine despite Mansfield's warnings that memory resident programs might "clash." Mansfield allows you to change the Personal REXX Interrupt (by default, 60H) through a DOS environment variable, so I suspect that Personal REXX will prove a well-behaved resident program.

I would recommend running Personal REXX off a ramdisk and having that as the first drive in your path. That way, the hard (or floppy) disk does not have to grind away searching before loading the REXX program. In this setup, REXX programs load instantly. If you write cpu-intensive tasks in REXX (such as sorting programs, or a program to find every occurrence of the word "the" in a large file), you might notice the processing time. For instance, Example 3 above iterates once for each power you calculate. It took REXX five seconds to tell me that 2 raised to the 1,000th power was 1.07150863E+301.

The Personal REXX package includes a windowing package (with an example using spreadsheet style menus), functions to read and write to the screen (including the ability to specify whether the BIOS or direct screen addressing should be used and whether "snow" should be accounted for), and several new utility functions. The package also includes an Applications Programming Interface (API) that allows a user application to invoke REXX and for the user to extend the range of functions in the REXX language. The distribution disk contains example interfaces to 8086 assembly language, Microsoft C, and Microsoft Pascal.


DEVELOPMENT TOOLS FOR C

- A COLLECTION OF "TOOL BOXES" FOR THE DEVELOPMENT OF POWERFUL C PROGRAMS
- DESIGNED TO BE USED BY BOTH NEW AND EXPERIENCED C PROGRAMMERS
- EACH TOOL BOX IS FULLY DOCUMENTED -- WITH EXAMPLES

SYSTEM TOOL BOX
- ACCESS THE POWER OF YOUR PC
- KEYBOARD & SCREEN MANAGEMENT
- GRAPHICS & MUCH MORE
- 164 PAGE USER'S GUIDE

LIB + SRC $35

UTILITY TOOL BOX
- GENERAL PURPOSE FILTERS
- UTILITY PROGRAMS
- GENERAL PURPOSE FUNCTIONS
- 86 PAGE USER'S GUIDE

LIB + SRC $25

WINDOW TOOL BOX
- COMPLETE WINDOW CONTROL
- BUILD WINDOW DATABASES
- FULL ERROR REPORTING
- 84 PAGE USER'S GUIDE

LIB $45 -- SRC $125

MENU TOOL BOX
- VERTICAL/HORIZONTAL MENUS
- CREATE MENU DATABASES
- FULL ERROR REPORTING
- 78 PAGE USER'S GUIDE

LIB $35 -- SRC $75

- HARDWARE -- IBM PC, TI PRO OR TANDY 1000
- COMPILER -- MICROSOFT™ C (3.00 & LATER)
- CALL OR WRITE FOR FREE INFO & DEMO PROGRAM
- TECHNICAL SUPPORT IS UNLIMITED & FREE
- NO ROYALTIES

FLEMING SOFTWARE
5741 Wagonwheel Trail
Castle Rock, CO 80104
(303) 688 - 3584

Number 118 on the Reader Service Card



Tech Help!
The Ultimate Electronic Reference Manual

Destroy your obsolete paper manuals!

Tech *Help!* is a unique hypertext encyclopedia.
Pop up a Megabyte of technical information while in your editor or debugger.

- DOS 3.3 functions and all ROM BIOS services -- every detail!
- Every known PC/XT/AT data structure, including FCB, PSP, boot sector, partition table, FAT, EXEC control block, and more!
- Keyboard scan codes, Extended ASCII keystrokes, ANSI.SYS escape codes, ASCII tables, BIOS data map, and MUCH more!
- 8088/8086/80286 Instruction Set
- LIM Expanded Memory Specification
- DOS, BIOS, and hardware information in one hypertext source!
- Same engine as in DOS Help! (PC Magazine "Editor's Choice")
- Uses 62K of RAM when resident; 1 Megabyte of compressed text occupies only 340K of your disk. Hard disk recommended.

Written by **Dan Rollins**,
author of the Norton Online Guide to OS/2.

Order now—check, money order, or MC/Visa
Only $89.95 + $4 S&H. Full money-back guarantee!
Toll free order line . . . ask for extension **1233**

800-358-4477
in California call 800-833-7355

Flambeaux
1147 E. Broadway #56, Glendale, CA 91205 (818) 500-0044

Number 206 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

Mansfield has recognized that books on REXX are not as readily available as books on other high-level languages and includes Cowlishaw's *The REXX Language* in their package. I would recommend that the beginner also get the official IBM user's guide to REXX (referenced at the end of this article), which is, contrary to form, an epitome of clarity.

**Personal REXX Applications**

That brings us to the question of what you might use Personal REXX for. As mentioned at the start, Mansfield's KEDIT can use REXX programs (called macros) to perform almost any task you could ask of an editor. Two other uses of Personal REXX are as a replacement for the DOS batch facility and second, as a language for applications beyond the power of the DOS batch facility.

As an example of the first, consider the steps you go through to invoke the Microsoft Macro Assembler (and most compilers). Typically, a DOS batch file contains a set of commands to assemble, link, and **exe2bin** the source file. The DOS batch facility cannot even prompt the user to ask whether he or she wants to **exe2bin**. In Personal REXX, a typical assembly language programmer's .REX file can do the following: First, check whether the file exists, then call the built-in REXX function to get the time; if all checks out, assemble the file and act on the return code, making sure to return the finishing time and the elapsed time. At the **exe2bin** stage, the REXX program prompts for a response and then branches on the result—either creating the .COM file and cleaning up the disk or going straight to the cleanup. Again, REXX could give the programmer timing statistics if desired. A simple program like this provides more functionality than the DOS batch facility and takes only about 15 minutes to write.

There are a multitude of examples of applications programs written in REXX (including several on Mansfield's bulletin board). An easy one that takes about five minutes to write is a utility to hide and unhide files. In an office environment this would be very handy. Just in case you forget the name of your hidden file, the REXX listfile utility can list out all files, hidden or not.

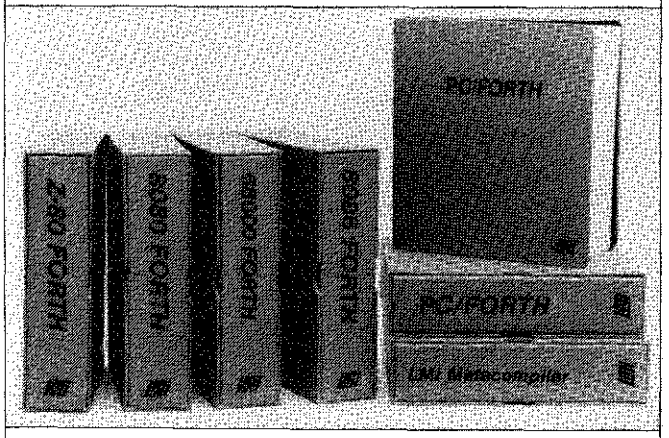
The following is a four function calculator written in REXX:

```
/* CALC.REX Simple REXX four-function calculator */
arg parm

if parm = '' then say 'Syntax is: "calc <expression>"'
   else interpret say parm
```

The program is invoked from the DOS command line with the command **calc** followed by the expression to be calculated. For example, **calc 18.261*7** produces the answer 127.827. The first line of the source code is a comment. The second line informs the interpreter that one argument has been passed to the program, which should be assigned to **parm**. The next line verifies that there is indeed an argument and, if not, informs the user of the correct syntax. The **else** statement uses the REXX interpret instruction to dynamically build a string that is then executed as though it had existed as a source line in the program.

A slightly more complicated example is a program called TRAVERSE supplied on the Personal REXX distribution disk. Move over, Peter Norton—this thing looks like it came straight from the "Chuck Norris Utilities." TRAVERSE goes through every directory on a given disk and performs the same command. The obvious application is deleting those infernal .BAK


**TOTAL CONTROL**
**with *LMI FORTH*™**




**For Programming Professionals:**

**an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers**

**For Development:**
**Interactive Forth-83 Interpreter/Compilers**
- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- *Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support*

**For Applications: Forth-83 Metacompiler**
- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- *Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303*
- No license fee or royalty for compiled applications

**For Speed: CForth Application Compiler**
- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

**Support Services for registered users:**
- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

***Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.***

***Laboratory Microsystems Incorporated***
*Post Office Box 10430, Marina del Rey, CA 90295*
*Phone credit card orders to:* ***(213) 306-7412***

**Overseas Distributors.**
*Germany:* Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
*UK:* System Science Ltd., London, 01-248 0962
*France:* Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
*Japan:* Southern Pacific Ltd., Yokohama, 045-314-9514
*Australia:* Wave-onic Associates, Wilson, W.A., (09) 451-2946

Number 63 on the Reader Service Card


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

files that so many programs create. Just enter **traverse erase *.bak**, and every single .BAK file is erased.

At a slightly more advanced level, how about a program that automatically keeps a record of your computer usage (remember all those IRS regulations)? Let's call it REXXLOG. The first time you run REXXLOG, it creates a log file as shown in Figure 1. Every time you log on, REXXLOG automatically saves a record of your logon time in a global variable on disk. When you finish, you type LOGOFF, and the starting time, finishing time, and elapsed time are appended to a usage log file as a new record thus:

**Figure 1**

REXXLOG--THE REXX COMPUTER USAGE LOG
Version 1.0 Copyright John A. Doe, 1987. All Rights Reserved.

| Date | Time Logged on | Time Logged off | Total Time On |
|---|---|---|---|
| Friday 1 May 1987. | On at 03:56:11. | Off at 03:56:22. | Time on: 00:00:11. |
| Friday 2 May 1987. | On at 04:35:13. | Off at 06:31:04. | Time on: 01:55:51. |

Of course, every now and again you get logged off involuntarily by power outages, etc. In this case, REXXLOG is smart enough to know that it should not update your usage log as you will be logging on straightaway. The only manual input you have to remember is to type **logoff** when logging off in order to update the usage log. REXXLOG uses the file to count your total time on during the month and displays the total time when you log on or log off. At the end of each month, REXXLOG changes the logfile name to archive it and starts a fresh file.

This is a medium-sized REXX application, yet it illustrates several aspects of the REXX language design philosophy. First, it is structured as several small subroutines, illustrating that REXX is a structured language. Second, since REXXLOG performs screen and text file I/O, it shows REXX string manipulation routines in action. Third, it illustrates the use of the REXX stack. In this case we use the stack as a temporary storage area for the file records of computer usage in preparation for summing the total system time on over the month. Finally, it would take a relatively inexperienced REXX programmer about one evening to write REXXLOG—such is the straightforwardness of the language.

## Summary

Before I used Personal REXX, I had used REXX extensively in a mainframe environment. In the mainframe world, even people who are otherwise averse to programming become REXXaholics. Everyone who tries it seems to love it. The language is inherently intuitive, so programming quickly yields tangible results. I was always baffled that IBM had not chosen to distribute a REXX interpreter in lieu of the BASIC interpreter with PC-DOS. Fortunately, Mansfield Software has stepped into the breach with "the program that IBM should have written," and they've done a fine job.

## References


Abacus, A. "REXX: A Beginner's Alternative." *Computer Language* 3, No. 6 (June 1986).

Cowlishaw, M.F. *The REXX Language*. Englewood Cliffs, N.J.: Prentice-Hall, 1985.

IBM Corp. *System Product Interpreter User's Guide*. New York: IBM Corp., SC24-5238-1. Release 3, February 1984.

Mansfield Software Group. *Personal REXX User's Guide*. Storrs, Conn.: Mansfield Software Group, Version 1.60, 1987. ♦

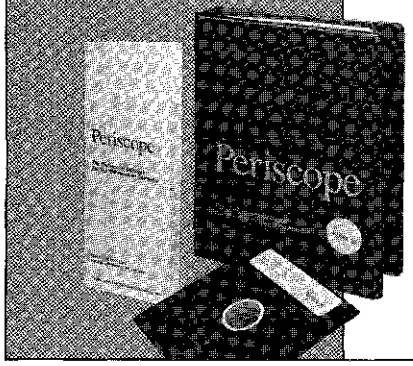

**Personal REXX V1.6.**
Mansfield Software Group
P.O. Box 532
Storrs, CT 06268
(203) 429-8402
$125.00 + $3 shipping.

One diskette (5.25-inch or 3.5-inch). Not copy protected. Requires IBM PC/XT/AT with 256K (384K recommended), DOS 2.1+, one floppy or 3.5-inch drive. Supports LIM EMS 3.2. Warranty: Mansfield will replace if REXX is incompatible with your equipment. License Agreement: You may use REXX on more than one machine, so long as it is not being used on more than one machine simultaneously.


*Andrew J. Chalk, Ph.D., is President of Magna Carta Software, P.O. Box 475594, Garland, TX 75047. Magna Carta's latest product is C Windows Toolkit, a windowing, menuing, and video control library for C programmers.*


*Code follows.*


The New Periscope Version 4




***...Gives you all the right stuff for debugging! No matter which model you pick, you have the same powerful software to help you track down hard-to-find bugs fast.***

***New in Periscope Version 4:***

- View local symbols from Microsoft C
- Debug Microsoft Windows applications
- Set breakpoints in PLINK overlays
- Improved source-level support
- Monitor variables in a Watch window
- 80386 debug register support
- Debug using a dumb terminal
- PS/2 watchdog timer support
- Use mixed-case symbols

*"The best keeps getting better... Once again, Periscope has raised the industry standard for debuggers!"*

**—David Nanian**
**BRIEF co-author**

**Periscope I** includes a half-length board with 56K of write-protected RAM; break-out switch; software and manual for **$455**.

**Periscope II** includes break-out switch; software and manual for **$175**.

**Periscope II-X** includes software and manual (no hardware) for **$145**.

**Periscope III** includes a full-length board with 64K of write-protected RAM, hardware breakpoints and real-time trace buffer; break-out switch; software and manual. Periscope III for machines running up to 10 MHz is **$1395**.

***Call Toll-Free for free information or to order your Periscope today!***

MAJOR CREDIT CARDS ACCEPTED. **800-722-7006**

The Periscope Company, Inc.
1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
404/875-8080

Number 6 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PJ CODE
All code in this issue is available on 360K, MS-DOS disk, $10.00 postpaid. ORDER BY PHONE — VISA/MC orders call toll free 800/234-0386. Mail order information on page 89.

## Code from "The REXX Language" by Andrew J. Chalk

1

```
REXXLOG
/* Copyright Andrew J. Chalk, 1987, 1988. All Rights Reserved.              */
/* Purpose: A utility to create and maintain a log computer usage           */
/* Usage: i)   Install RXINTMGR, STACKMGR, RXBATMGR,                        */
/*             ANSI.SYS and GLOBALV.                                        */
/*        ii)  Ensure EXECIO.EXE is in your path when LOGOFF is called.     */
/*        iii) Place the command "Logoff on" in your autoexec.bat           */
/*        iv)  When you finish work, type "logoff"                          */
/* REXXLOG will create a file called USAGE.LOG as specified below           */
/* This file contains a record of your computer usage.                      */
/* First version 5/15/87. Last update: 05-11-88.                            */

trace n
options newcom

/********************* VARIABLES GLOBAL TO THE PROGRAM **********************/
/* USERS: This is where you record the name of your logfile */
logfile = 'c:\rexx\myrexx\usage.log'             /* location of log */
ver = 'Version 1.2'                             /* program version number */

/*****************************************************************************/


arg status .                            /* are we logging on or off?   */
  'cls'
  call intro                            /* Show the intro screen       */
  select
    when status = on then call logon    /* You want to logon..         */
    when status = '' then call logoff   /* You want to logoff..        */
    otherwise call error 1
  end
  call total_time                       /* Let's find out how long     */
  if dosdir('c:\lasting.bak') <> '' then call dosdel 'c:\lasting.bak'
exit(0)


/* THIS PROCEDURE PUTS AN INTRODUCTORY HEADER ON THE SCREEN */
intro : procedure expose ver

  h1 = 'REXXLOG -- THE REXX COMPUTER USAGE LOG'
  cpyrt = right('(C) Andrew J. Chalk, 1987,1988. All Rights Reserved',80)
  h2 = overlay(ver,cpyrt)
  say Centre('33;40;1m' h1 '0m',96)
  say h2
return


check_file : procedure expose logfile ver       /*  create a new log?  */

  if dosdir(logfile) <> '' then call check_date
  if result = 1 then return
  h1 = Center('REXXLOG -- THE REXX COMPUTER USAGE LOG',80)
  cpyrt = right('Copyright Andrew J. Chalk, 1987. All Rights Reserved.',80)
  h2 = overlay(ver,cpyrt)                       /* what an egotist! */
  t1 = left('Date',80)
  d1 = 'On Time'
  t1 = overlay(d1,t1,23)
  d2 = 'Off Time'
  t1 = overlay(d2,t1,32)
  d3 = 'Purpose of work'
  t1 = overlay(d3,t1,41)
  d4 = 'Time On'
  h3 = overlay(d4,t1,73)
```

2

```
  call lineout logfile, h1
  call lineout logfile, h2
  call lineout logfile, ' '
  call lineout logfile, h3
  call lineout logfile
  say Centre('33;40;1m File' logfile 'not found. New Logfile Created. 0m',95)

return


check_date:procedure expose logfile       /* Has the month changed?    */
                                          /* If so, create an archive */
    file_month = substr(dosdir(logfile,d),1,2) /* the month of the logfile */
    actual_month = substr(date('U'),1,2)       /* the month at runtime */
    if actual_month = file_month then return 1
    fpath = substr(logfile,1,lastpos('\',logfile))
    fspec = dosdir(logfile,n)
    if pos('.',fspec) <> '' then fname = substr(fspec,1,pos('.',fspec)-1)
    else fname = fspec

    /* HOW TO USE SETS IN REXX */
    monthset = jan feb mar apr may jun jul aug sep oct nov dec
    month = word(monthset,file_month)
    'copy' logfile fpath||fname'.'month '> nul'
    'erase' logfile
return 0


logon : procedure expose logfile       /* logon procedure */

    call check_file    /* if logfile does not exist then create a new one */
    if dosdir('c:\lasting.glv') <> '' then return
    reason = purpose()
    day = left(date('w'),9)
    fdate = right(date(),11)
    info = day fdate'. On at' time()
    'globalv select worklog setpl instring1' info
    info = day fdate'. On at' time() 'to work on' reason
    'globalv select worklog setpl instring2' reason
    say Centre('33;40;1m The Following Logon Information was Recorded by
     REXXLOG 0m',95)
    say Centre('7m' info ' 0m',88)

return


purpose: procedure

    say '1mEnter purpose of logging on (for tax records):0m'
    say
    parse pull reason
    reason = strip(substr(reason,1,80),t)

return reason


logoff : procedure expose logfile

    call check_file
    if dosdir('c:\lasting.glv') = '' then return  /* already logged off! */
    'globalv select worklog get instring1'
    intime  = word(instring1,words(instring1))
    outtime = time()
    time_on = elapsed_time(intime,outtime)
    'globalv select worklog get instring2'
    temp1 = left(word(instring1,1),9)     /* the day */
    temp2 = left(word(instring1,2),2)     /* the date */
    temp3 = left(word(instring1,3),3)     /* the month */
```

*Code continues . . .*



PRODUCED © 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Chalk Code, Continued

3

```
    temp4 = left(word(instring1,4),4)     /* the year */
    temp5 = left(intime,8)
    temp6 = left(outtime,8)
    temp7 = left(instring2,31)
    temp8 = left(time_on,8)
    b3 = ''
    outinfo1 = temp1 temp2 temp3 temp4 temp5 temp6 temp7 temp8
    outinfo2 = instring1'. Off at' outtime'. Time on:' time_on
    outinfo3 = 'Work:' instring2
    call lineout logfile, outinfo1
    if result <> 0 then call error result
    call lineout logfile
    say Centre('33;40;1m The Following Logoff Information was Recorded by
     REXXLOG 0m',95)
    say Centre('7m' outinfo2 '0m',88)
    say Centre('1;44;35m' outinfo3 '0m',94)
    if dosdir('c:\lasting.glv') <> '' then 'erase c:\lasting.glv'

return


elapsed_time:procedure                          /* How long were we on? */
arg start, finish
ehours = 0;emins=0;

    shour = substr(start,1,pos(':',start)-1)
    smin = substr(start,pos(':',start)+1,2)
    ssec = substr(start,7,2)
    fhour = substr(finish,1,pos(':',finish)-1)
    fmin = substr(finish,pos(':',finish)+1,2)
    fsec = substr(finish,7,2)

    start_secs = (shour*3600) + (smin*60) + ssec
    finish_secs = (fhour*3600) + (fmin*60) + fsec

    esecs = finish_secs - start_secs
    if esecs < 0 then esecs = esecs + 86400     /* midnight passed */
    if esecs > 3600 then do
        ehours = esecs % 3600
        esecs = esecs // 3600
    end
    if esecs > 60 then do
        emins = esecs % 60
        esecs = esecs // 60
    end
    if ehours < 10 then ehours = '0'ehours;
    if emins < 10 then emins = '0'emins
    if esecs < 10 then esecs = '0'esecs
    etime = ehours':'emins':'esecs

return etime


total_time:procedure expose logfile /* How long were we on this MONTH!*/

    thours = 0;tmins=0; tsecs=0;

    stack_state = stackstatus() /* is the stack installed and large enough? */
    parse var stack_state flag id free
    if flag = n | flag = d then call error 3 /* stack not installed */
    filesize = word(dosdir(logfile),2)
    if filesize >= free then call error 4
    theirs = queued()


    address command 'execio * diskr' logfile '0 (fifo finis notype'
    if rc < 0 | rc = 28 then call error rc
    do 4                                    /* pull header */
```

4

```
        pull
    end

    do while queued() > theirs
        pull line
        time = word(line,words(line))       /* the time on is the last line */
        shour = substr(time,1,pos(':',time)-1)
        smin = substr(time,pos(':',time)+1,2)
        ssec = substr(time,7,2)
        secs = (shour*3600) + (smin*60) + ssec
        tsecs = secs + tsecs
    end
    if tsecs > 3600 then do
        thours = tsecs % 3600
        tsecs = tsecs // 3600
    end
    if tsecs > 60 then do
        tmins = tsecs % 60
        tsecs = tsecs // 60
    end
    if thours < 10 then thours = '0'thours
    if tmins < 10 then tmins = '0'tmins
    if tsecs < 10 then tsecs = '0'tsecs
    ttime = thours':'tmins':'tsecs
    say Centre('36;47;1m Total Time On This Month is' ttime' 0m',94)

return ttime


error: procedure                                /* Errors happen! */
arg enum

select
    when enum = -8 then,
      say '30;41m Error: 31;40m Insufficient REXX storage run command.0m'
    when enum = -3 then,
      say '30;41m Error: 31;40m Requested program could not be found. 0m'
    when enum = 1 then,
      say '30;41m Error: 31;40m Incorrect syntax. 0m'
    when enum = 2 then,
      say '30;41m Error: 31;40m Could not write line to'upper(logfile)||'. 0m'
    when enum = 3 then do
      say '30;41m Error: 31;40m Stack manager disabled or not installed. 0m'
      exit(-1)
    end
    when enum = 4 then,
      say '30;41m Error: 31;40m Insufficient REXX stack space. 0m'
    otherwise say '30;41m Error: 31;40m Unidentified error. 0m'
end
call lineout logfile
exit(1)
```

◆



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## Program Editors and Tools

**MKS Toolkit Version 2.3**
Mortice Kern Systems, Inc.
35 King Street North
Waterloo, Ontario
Canada N2J 2W9
(519) 884-2251

Mortice Kern Systems is representative of a number of quality software houses in the Toronto area with strong roots in the Unix world. The MKS Toolkit ($169) is a powerful collection of utility programs designed to emulate the Unix shell and user system under DOS. The toolkit supports over 115 commands compatible with Unix System V3. Among the programs included, the most significant are the MKS Korn Shell, vi editor, awk, ed/sed, yacc, and diff, the grep utilities, and the sort/merge, spell, cpio, and pack/unpack utilities. The central component is the Korn Shell, an implementation of the latest Unix shell having full compatibility with the earlier Unix Bourne shell along with enhancements from the Unix C shell. With the support of the accompanying utilities, powerful Unix shell scripts can be run under DOS. The shell supports command aliasing, both vi and emacs style editing modes, a previous history command, conditionals and flow control, shell variable expansion, and arithmetic evaluation. A few pieces are missing. The help utility is of little use since the manual pages are not currently distributed on DOS.

The toolkit supports four different configurations. The most limited is to use DOS (command.com) as your default shell and occasionally use the MKS Korn Shell (sh.exe). The MKS utilities may be used under either command.com or the MKS shell. A batch file is provided to change the names of four built-in DOS shell commands so that the more powerful MKS toolkit versions may be substituted. The Korn Shell would be invoked only when needed. Any normal DOS program may be invoked from the Korn Shell, but TSR programs are best invoked before the shell since memory fragmentation can occur. The second configuration still uses command.com but invokes sh automatically via an autoexec.bat file. The Korn shell in these configurations takes about 90K of memory, but a user can exit back to DOS's command.com at will.

The last two configurations substitute sh.exe for command.com as the default command interpreter in your config.sys file. These configurations consume less memory since the 40–50K normally required by command.com is now used. The only shell would now be the MKS shell. The last configuration is the closest to Unix that substitutes init.exe as the command shell, which then invokes the Korn shell. In this scheme, login and passwords are supported.

The toolkit is distributed on either five 360K disks or fewer 3.5-inch disks. Documentation consists of a single spiral bound manual that includes a brief installation guide, a general overview (72 pages), and Unix style manual pages for all of the commands. This large command reference section (300+ pages) has a standard Unix-style table of contents and permuted word index. Both vi ($75) and awk ($75) are sold as separate products. The toolkit's awk is a small model version while the separate version of awk includes both small and large memory models and optional 8087 versions. Customer support is provided free for the first 30 days (phone, mail, E-mail, BIX, and Usenet). Preferred customer support costs $79 per year and includes a toll-free 800 number.

I was able to download some shell archives straight off Usenet and run them transparently under the Korn Shell as if I had a Unix machine. Clearly, this package has a lot of power going for it. The MKS toolkit can be used as a decent substitute for the real thing (Unix) under DOS.

**BRIEF 2.1**
Solution Systems
541 Main Street, Suite 410
South Weymouth, MA 02190
(800) 821-2492

Solution Systems has released a new version of BRIEF, their Basic Reconfigurable Interactive Editing Facility ($195). The choice of an editor is a very personal decision often based on previous experiences with earlier products and styles. Some programmers prefer small, quick editors with limited features while others favor powerful editors that in themselves approach programming languages. For some time BRIEF has been considered the premier programmer's editor in the DOS world, the editor against which others are compared. BRIEF supports editing multiple files in multiple windows and includes an online help and powerful search and replace ("translate" in BRIEF jargon) commands using regular expressions. If I had to single out only a handful of features, they would be BRIEF's Undo feature, its powerful macro language facility, and its smart indenting and syntax checking features for editing files, particularly C files. BRIEF's Undo supports undoing most previous commands—the default is 30, but as with many of BRIEF's features, the number is configurable. The Undo command is stack based, so you must undo the last 20 commands to get to the 21st one. The macro language supports the creation of powerful customized commands that may be invoked automatically based on the extension of the file you're editing (C, ASM, etc., for example). The default Editor keystroke commands are generally mnemonic. The commands make extensive use of Alt, Ctrl, Shift, and the function keys. A *keys* command (poorly documented) does allow rebinding a different keystroke to any command although no alternate key bindings are provided.

BRIEF is distributed on two 360K disks, a program disk and a utility disk with a SETUP program (3.5-inch format disks are also available when ordering or with a mailer). The SETUP program is only briefly (no pun indented) mentioned in the manual, but it is generally menu based and easy to use. SETUP will install the program on floppy disks or a hard disk, compile various macro files, create optional autoexec.bat and config.sys files, and even patch command.com (DOS 2.x) to allow a larger environment space. BRIEF makes extensive use of the DOS environment to set various options for editor use. These options include video and multitasking (DESQview and Taskview) device drivers, display features (screen size and window borders), keyboard compatibility options and repeat rate, the expansion of tabs to spaces, ending files with Ctrl-Z, automatic file save and backup, cursor shapes, maximum undo commands, location of help files, macros and macro packages to load, and even a language compiler and compiler switches to use when compiling from the editor. Some of these options may also be invoked on the command line. Macros are compiled before use with the editor. Sample macro language packages support several types of indenting: regular, smart indenting (recognizes C statements like *if* to change the indent level), and template indenting (smart indenting for C with actual statement completion). Version 2.1 adds the option of saving the edit environment to disk to free more memory for a compile invocation from the editor. BRIEF will automatically locate errors after a compile step but will move only forward through the error list.

BRIEF is a virtual editor—files do not have to fit into memory. Although the limit is not documented, the current version is limited to files of fewer than 64K lines. This shouldn't pose a problem for most users. I was able to edit a 2.5-megabyte file (35,000 lines). BRIEF has limits on the line length of files to be edited. The editor can't handle files ending only in linefeed (Unix files, for instance)—they are read in as a single line. The editor keystroke interface can be customized, but some keys are permanently bound and can't be changed. Those weaned on WordStar or a Borland editor environment can't easily create a WordStar-like keyboard interface. The macro language is based on a Lisp-like syntax (parentheses grouping and


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

reverse Polish evaluation) and although powerful may be difficult for many DOS programmers to learn. If you cut your teeth on emacs and Lisp, you'll have no problem.

Documentation is a *User's Guide* (161 pages) and a *Macro Language Guide* (195 pages), both with an index. The *User's Guide* contains tutorial sections and a Command Reference Guide. The software is supported via phone and a 24-hour bulletin board.

**PI Edit Version 2.00**
The Iliad Group
2697 Union Street
San Francisco, CA 94213
(415) 563-2053

The Iliad Group's PI ($195) is a relatively new programmer's editor for DOS, another powerful, feature-rich editor for editing multiple files in multiple windows. PI is intended to provide a complete development environment. If I were to mention only a few of PI's strengths, they would include the powerful macro capability, syntax checking, the enhanced undo facility, and some interesting find/replace features. The PI macro language (PML) is C-like and easy to learn. Macros are compiled before use, and there is support for debugging. The macro compiler is an external program, but it is usually invoked directly from the editor. The Undo feature allows undoing any single deletion without processing an entire stack of commands. A convenient help system is available via pop-up menus. Most editing parameters are customizable, and the key-board is mostly configurable—a few keys are permanently bound. Two keyboard bindings are provided—the default PI binding and a Wordstar-like binding. Aside from the normal cursor pad one-keystroke commands, most commands are two-keystroke sequences—a lead-in mnemonic key (say Ctrl-W or Alt-W for window commands depending on the binding) followed by a second menu pick key. As a form of help similar to WordStar's, a pop-up command menu appears for the selection if the second key stroke is delayed. About 200 different commands are built in. Windows are overlapping and may slide around the actual screen. The number of backup files to maintain is configurable. A powerful *findall* command allows copying all lines with a particular string match into a new window. The contents of this window may be edited and merged back into the original file with each line in its original location.

PI is not a virtual editor, but is limited by available memory—speed was one of the authors' main concerns (they are working on a virtual memory version). For editing larger files, PI does support both the earlier EMS 3.20 and the EEMS (AQA) expanded memory protocols. The editor consists of a 300K executable file (really lots of overlays) and when loaded leaves about 200–300K of memory for the files you may be editing. Editing operations and screen updates are fast. PI could not edit files ending only in linefeed (Unix) and wouldn't edit files bigger than 200K (without EMS memory). PI does support transparently editing EDCDIC record oriented IBM files.

*Continued . . .*


# MISOSYS

**SOFTWARE FOR YOUR P.C.**

**Order Now 800-MISOSYS (647-6797)**

## ED/ASM-86™

- 80X86 Assembly Language development

An integrated editor, assembler, linker, and debugger for 8086, 80186, 80286, and 8087 assembly and disassembly. All features are in a single EXE file.

- a full-featured line EDITOR with intra-line editing, block move and copy, partial save and load.
- a full-screen turbo-type editor is supported on CGA-equipped systems.
- the ASSEMBLER supports structured coding; can directly generate a COM or EXE file; assembly to memory for faster debugging.
- The DEBUGGER includes symbolic disassembly of arbitrary code to disk.

ED/ASM-86 [M-86-030] . . ~~$124.95~~ + $5 S&H

**NEW LOW PRICE: $99.95**

## RATFOR-86™

- Rational FORTRAN translator

A professional implementation of RATional FORtran by James Beard, Ph.D. Provides structure and greater portability to FORTRAN programs to reduce your programming time and effort dramatically over that required when writing directly in FORTRAN. Includes tutorial reference manual.

- compiler has extensive macro support
- "arith" macro for binary arithmetic
- short form READ and PRINT
- enhanced switch case support
- Requires a FORTRAN compiler

RATFOR-86 [M-86-73] . . . $99.95 + $5 S&H

**MISOSYS Inc.**
P.O. Box 239
Sterling, VA 22170-0239
703-450-4181 MC, VISA, CHOICE
800-MISOSYS/647-6797

VA residents add 4½% sales tax.
S&H: Canada add $1;
Foreign use S&H times 3

Number 121 on the Reader Service Card

# The Heap Expander™ version 2.0

Now your programs can have virtually unlimited heap space using expanded memory, extended memory, disk space, or any combination of the three. And it's all transparent. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still $59.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 and 5.0
- Turbo Pascal 3.0 and 4.0

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

*Idaho residents add 5% sales tax
Foreign customers add $4.00 for shipping and handling.

**The Tool Makers**
75 Mt. Hermon Rd.
Scotts Valley, CA 95066
(408) 438-7412

Number 166 on the Reader Service Card

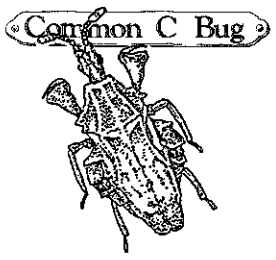

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# T-SHIRTS

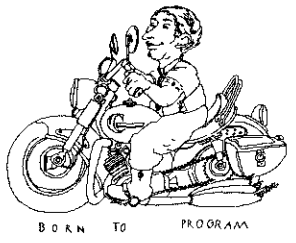
**$12.00 each** -- includes shipping & handling

Multi-color imprints on the highest quality 50/50 cotton T-shirts.

SIZES S, M, L, XL




COMMON C BUG




BORN TO PROGRAM

PARALLEL PORTS

RS-232 HEAD

**BUGS RULE** 4 color imprint on a white shirt

# PJ ISSUE SPECIAL!

**$14.95** REG $21.00

## BACK ISSUE SPECIAL!

**5.1 Jan/Feb 87**
**5.2 Mar/Apr 87**
**5.3 May/Jun 87**

The Ray Duncan Series—
**Starting Your Own Software House**
Regularly **$21.00**, these 3 popular back issues can now be yours for just **$14.95.**

Offer expires November 1, 1988

# BACK ISSUES

**$7.00 each** PREPAID ONLY

A few of the Back Issues currently available

**Vol. 4.6:** DTACK GROUNDED; Unix Books, Part 1; How many amoebas at one o'clock; DOS functions 45H and 46H; DOS Terminate ans Stay Resident; Turbo Pascal inline code; Pascal—Assembler source code for DOS function calls; Alignment problems in C and Managing memory allocation; Proposed ANSI C Language Standard, Part 4; The Canon A1 laser printer; Genifer dBASE applications generator reviewed; Advanced Trace86 debugger reviewed.

**Vol. 5.1:** DTACK GROUNDED; The new DOS; Marketing and rules to live by; File redirection; Inside the EGA; String allocation in Pascal; ANSI C Language Standard, Part 5; More tips on formatting BASIC data.

**Vol. 5.2:** DTACK GROUNDED; Starting your own software house; Pink noise; TSR graphics print screen utility; Coding guidelines for resident programs; Parallel processing with the EGA; C—portability in practice and using header files; Proposed ANSI C Language Standard, Part 6; Indexing in BASIC; A first look at Turbo C.

**Vol. 5.3:** Starting your own software house--publicity; Unix books, Part 2; Consulting—merger mania; EGA data control; Modula 2 for Pascal lovers; Optimizing Turbo Pascal; ANSI C language standard; Coping with ANSI C and forcing file updates; Formatting BASIC output with buffered PRINT USING.

**Vol. 5.4:** Starting your own software house—technical support; Unix books, Part 3; Consultant productivity and effectiveness; EGA set/reset capabilities; ANSI C language standard; Character tables and string functions in C; MWWIND—a menu with windows utility; Variable passing and hiding in QuickBASIC 2 subprograms.

**Vol. 5.5:** Intel's 80386 vs. OS/2; Metagraphics President Jack Davis interview; Consulting—making computer tools pay for themselves; ASM—the invisible CapsLock; TSRs using expanded memory with Turbo Pascal; Note on optimizing Turbo Pascal; Proposed ANSI C Language Standard; Printer output from Turbo C programs; Matchup of Turbo Basic 1.0 and QuickBASIC 3.0.

**Vol. 5.6:** Multics, OS/2, paging and segmentation; The Usenix conference and other news; How to get rich in consulting; Shell in Marshal Pascal; Turbo Pascal 4.0 preview; Handling fixed length character arrays in C; The proposed ANSI C Language Standard.

**Vol. 6.1:** Consulting is an attitude; Write mode 3 of the VGA; The LIM 4.0 expanded memory specification; Environment variables in Turbo Pascal 4.0; A timer program in C; Porting mainframe software to the PC--FORTRAN compilers under DOS and the Definicon 32-bit coprocessing engines.

**Vol. 6.2:** Hal Hardenbergh on RISCs, CISCs, and Fabs; Pink noise; Yet another VGA write mode; Inside Turbo Pascal 4.0; The proposed ANSI C Language Standard; Converting a Turbo C program to a TSR; 80286 and 80386 programming; 80386 software development; The Norton Guide for BIOS/DOS Assembly.

**Vol. 6.3:** Hal Hardenbergh on memory trends; Pink noise; Getting your consulting tools in order; Gordon Letwin on OS/2 memory management; Virtualizing the 386; Pascal—A CASE of the jumps; Turbo Professional 4.0; BASIC—CALL, GOSUB, and multi-line functions

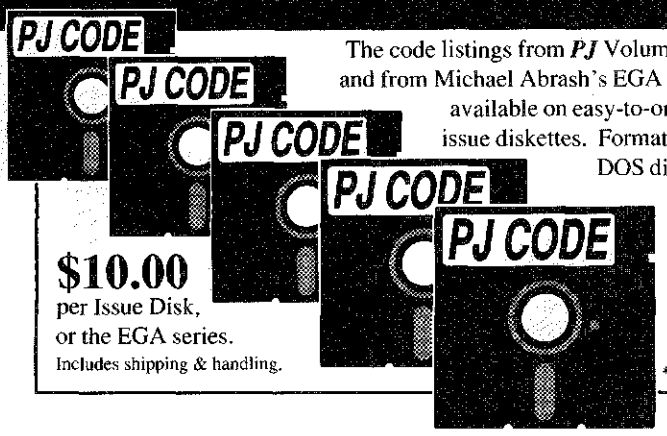
**Vol. 6.4:** Job-shopping; Windowing systems overview; Reading VGA memory; X Window—a view from DOS; A digital clock in C for MS Windows; Windowing in TopView, DESQview, & MS Windows; MS C 5.1 and the ANSI Standard; A mouse unit for Turbo Pascal 4.0.

10% discount on any order of 10 or more T-Shirts, Disks or Back Issues.

All mail orders must be accompanied by check or Money Order -- PREPAID ONLY.

# CODE DISKS

Use the order form on the card opposite.
Order by phone with your VISA/MC.
CALL TOLL FREE
**800/234-0386** ext 53




The code listings from *PJ* Volume 5.5 forward, and from Michael Abrash's EGA series are now available on easy-to-order individual issue diskettes. Formatted 360K MS-DOS disks standard.*

**$10.00** per Issue Disk, or the EGA series. Includes shipping & handling.

* Other formats available

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

PI has screen and keyboard compatibility if operating under Microsoft Windows. PI supports most compilers—those that can redirect line numbers to standard output to error files. The list of currently supported compilers includes a dozen C products, three 80x86 assemblers along with at least one vendor's C++, Pascal, FORTRAN, COBOL, dBASE, SQL, and BASIC compilers. The compiler error environment is far superior to BRIEF's. Errors are highlighted, and you can move both backwards and forwards in the error list and source file. The editor may also be used for stream editing as a filter. The program comes on two disks with a single, large spiral-bound manual (300+ pages). The manual could be improved, but with the excellent help system, I found PI easy to learn and use.

## Debuggers

**Periscope Version 4**
The Periscope Company
1197 Peachtree Street
Atlanta, GA 30361
(404) 875-8080

Although the software debugging period can equal the entire time taken up by product design and coding, little is written about debugging. Programmers invest lots of time and money in compilers and other development tools, but debugging tools usually come last. Although Microsoft's CodeView symbolic debugger may get the most trade press, if you talk with many professional programmers you'll find they use debuggers like Periscope. The Periscope Company has been selling one of the best professional debuggers in the DOS world for some time. Periscope has always had a reputation as an excellent debugger for assembly language, and Periscope Version 4 includes features to make it an exceptional source-level debugger. The list of new features includes Microsoft Windows debugging support, using Microsoft C local symbols (CodeView), PLINK overlays support, a full screen terminal driver when using a serial port for debugging, use of 80386 debug registers if available, 80386 register display, watch windows, mixed case and long symbol names (32 characters), and PS/2 timer support.

Very often the problem with a new program is that it dies somewhere with no clue—in some near endless loop. Other common debugging problems under DOS occur with heavily overlaid programs (commonly using Phoenix's PLINK linker) or with an errant program actually trashing the code or data used by the operating system or the debugger. Periscope comes in four versions, three with hardware assist, targeted at these and other problems under DOS. Model III, the top of the Periscope line ($1195), is a hardware/software product that includes a board with 64K of write-protected RAM (for the debugger code and data) that supports hardware breakpoints, a real-time trace buffer, plus its own program-interrupt "break-out switch." Model I ($345) is a scaled down version with a write protected RAM board and break-out switch. The two versions with write-protected RAM protect Periscope's software from programs that overwrite low memory, where the debugger normally resides. And since Periscope uses the board's memory, it doesn't compete for memory with the program being debugged. Model II ($175) incorporates just a simple break-out switch that can be installed without taking a slot. And Model II-X ($145) is just the software only debugger. I found the break-out switch an extremely useful feature of the Model II version. It allowed breaking out of some errant programs that were off in deep space to determine where they were looping or stuck.

The package consists of the primary debugging software plus a series of powerful tools and utilities to assist your efforts. They include a tool to generate symbol files from MAP and/or EXE files from PLINK or Microsoft LINK; a tool to convert local assembly language symbols to publics for debugging, reading, and saving the interrupt vector table; a tool to load any COM file as a device driver; plus a utility to clear the non-maskable interrupt (used by the break-out switch). It is even possible to change the symbol table loaded in Periscope when you're debugging GEM applications.

Periscope software allows for symbolic debugging of Microsoft Windows applica-


**MetaWINDOW - Power Graphics for your PC!**

PC TECH JOURNAL
"Product of the Month"
"... a technological tour de force for fast PC graphics."

NO ROYALTIES!

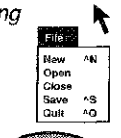
COMPUTER LANGUAGE
Oct '87 - Product Review
"...surprisingly powerful for the cost!"
Independently benchmarked the fastest of all graphic systems reviewed - faster than Halo, over 3 times faster than GSS*CGI!

MetaWINDOW is the advanced, high performance graphics development toolkit which bridges the gap between low level graphic primitive libraries and pre-packaged window managers. MetaWINDOW includes an expanded set of graphic drawing functions, plus the added functionality and performance required for designing multi-window desktop applications.

"MetaWINDOW is the best graphics package I've seen. Since I bought my PC, I've spent over $2500 on software. I've bought bad software, mediocre software, and good software. There's not much software I call excellent, but MetaWINDOW is excellent!"
J. Kayser
CMSI
Portland, OR

Unmatched Features!
- Display multiple bitmap or "filled-outline" fonts.
- Face fonts for bold, italic, underline or strike-out stylings.
- Full 8 function "rasterOp" transfers for writing, erasing, rubberbanding or dragging: lines, text, icons, bit images and complex objects.
- Create pop-up menus, windows and icons.
- Supports 70+ graphics displays including PS/2 VGA & MCGA.

- auto-cursor tracking
- pull-down menus
- pop-up windows
- comprehensive graphic functions
- unlimited fonts




10 Point 12 Point
Bold Italic

MetaWINDOW comes complete with language bindings for 20 popular C, Pascal and Fortran compilers, plus dynamic run-time support for over 70 graphics adaptors and input devices.

**MetaWINDOW**
Advanced Graphics Development Toolkit
4 disks, 3 260 page manuals - $195*

**New - MetaWINDOW/Premium**
Power graphics for new high performance TI34010 based graphics cards! - $495*
(Licensing required for redistribution of MetaWINDOW/Premium)
* Plus $5.00 shipping and handling

TO ORDER CALL 1-800-332-1550
For information or in CA call 408-438-1550

METAGRAPHICS
SOFTWARE CORPORATION
269 Mount Hermon Road
Scotts Valley, CA 95066

Number 143 on the Reader Service Card


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

tions, DOS device drivers, non-DOS programs, hardware interrupts, memory-resident programs, programs with overlays, and programs running under other environments like TopView or GEM in addition to your normal DOS programs. Along with monochrome and color support, Version 4 supports both EGA and VGA screens including 43- and 50-line modes, screen swapping, and dual monitor configurations. One of the more interesting features is a software trace buffer—the Model III hardware also supports a hardware trace buffer. The trace buffer is used to save the machine registers after execution of an instruction (up to 2016 entries). When displayed, the buffer shows the registers and a symbolic disassembly of the instructions saved. Periscope really comes into its own when tracking down those difficult to find bugs.

The Periscope software comes on a single 360K diskette with a solid manual (250+ pages). All products come with a money-back guarantee. Registered users receive free technical support and a free software update. A lower cost model of Periscope may be upgraded to a more expensive version for essentially the difference in cost.

**MagicCV**
Nu-Mega Technologies
P.O. Box 7607
Nashua, NH 03060
(603) 888-2386

Nu-Mega Technologies' MagicCV ($199) is another useful debugging tool for 80386


# OMNIVIEW
## ROCK SOLID MULTITASKING!

Packed with the power you need, OMNIVIEW takes you beyond the limits of DOS.

- Time-slices multiple jobs
- Takes full advantage of EMS 4.0
- Optional support for '386 Systems
- Manages megabytes of programs
- Controls resident utilities
- Lets you cut and paste between programs

OMNIVIEW unleashes your PC & converts it into an easy-to-use, multitasking workstation. Whether you have an 80386 or 8088 based computer, OMNIVIEW will squeeze out every ounce of power.

Solid & bulletproof, it's 47K assembly language engine gives you all the power without added overhead or complexity. Just ask the reviewers.*

"Sunny Hill provides true multitasking for most PC applications. It performs remarkably well, causing almost no noticeable reduction in the performance of each program . . . The user manual is well organzied and easy to read . . . It is an excellent product."
PC Tech Journal Sept '87

"It does an amazing job of handling not only multitasking, but in many cases, concurrent processing as well. I was amazed at it's success in handling the "impossible" combinations I threw at it."
"It is one of the best designed & implemented programs I've ever worked with. It is also one of the most bulletproof."
"What sets it apart from the pack is the fact that it runs in a network environment."
Data Based Advisor Aug '87

OMNIVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:
(800) 367-0651
or send $89.95 + 3.00 S&H ($8.00 Intl.) to:

**Sunny Hill Software**
*PO Box 55278, Seattle, WA 98155-5278*
For more information call (206) 367-0650 *Omniview formerly Taskview
DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

Number 25 on the Reader Service Card



# NROFF/PC™

*The* **REAL** *Thing for* **DOS**

**NROFF/PC** is a complete text formatting system for *MS-DOS* systems. Including:

**NROFF** The *powerful* UNIX text formatter

**TBL** A tool to assist with the layout of tabular material in *Nroff* documents

**MM** A comprehensive *Nroff* macro package for preparing books and technical manuals

**NEQN** A tool for describing mathematical equations in *Nroff* documents

- All tools are a complete port from the AT&T Documentor's Workbench 2.0
- It's **Fast!** We've modified *Nroff* especially for DOS for lightning speed
- Supports any Dot Matrix printer and many laser printers
- Specially Priced At **$99**
- A complete *Troff* typesetting system is available **NOW** for LaserJet and PostScript printers on **MS-DOS** for *$695*, XENIX and Microport UNIX for *$795.*

Elan Computer Group, Inc.
410 Cambridge Ave., Suite A
Palo Alto, CA 94306
(415) 322-2450

*Visa and MasterCard Accepted*

*Unix is a tradmark of AT&T*
*MS-DOS and Xenix are tradmarks of Microsoft*

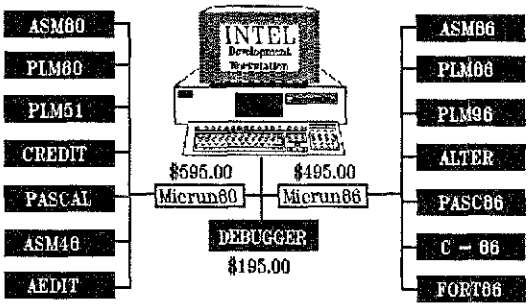
Number 173 on the Reader Service Card





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# RUN INTEL ON A PC




## DEVELOP and DEBUG

8 and 16 Bit INTEL CODE ON **YOUR** PC...WITH

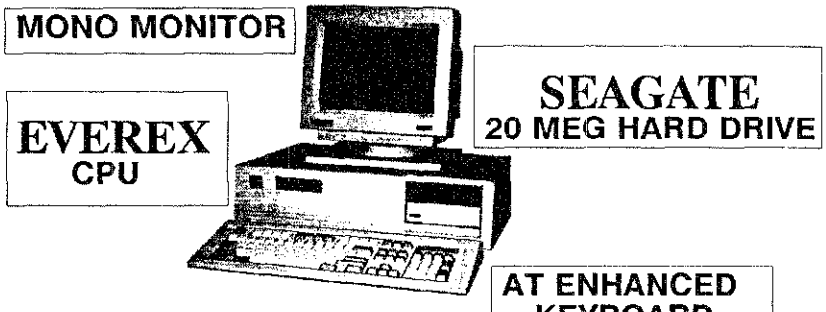
# MICRUN

**MICRUN 80**....8 Bit Development Tool (6 or 10 MHZ)
**MICRUN 86**.....16 Bit Development Tool
**8080 Debugger**.....Intel Type Software Debugger

- **Micrun 80/86 Package $ 895**
  *(includes Micrun 80,86 & 8080 Debugger)*

**********

**INTRODUCING**
**THE PC/INTEL DEVELOPMENT STATION**




**INCLUDES:**

- **Floppy Drive w/Controller**
- **MICRUN 80/86 PACKAGE**
- **MS-DOS & GW BASIC**
- **QUIKSAVE-Backup Utility Program**

| XT 10 MHZ V-20 | AT 12 MHZ 80286 |
|---|---|
| **$ 2,195** | **$ 2,995** |

All systems ready to go. Pretested (48 hr burn-in). Optional equipment availble.

**Call Today 1-800-888-8086**

Micro Interfaces Corp. 16359 NW 57th Ave., Miami, FL. 33014

*Intel, Everex & Seagate are registered trademarks of their respective companies.

Number **214** on the Reader Service Card


machines with extended memory. MagicCV is a software tool that allows Microsoft CodeView to run in extended memory on 80386 machines using only about 8K of conventional memory for operation. This allows the use of CodeView as a debugger on much larger programs (over 500K) than previously possible. MagicCV is like Nu-Mega's Soft-ICE 80386 debugger (see *PJ* 6.3) in that it truly uses the power of the 386 chip. MagicCV creates two 80386 virtual machines, one for CodeView in extended memory and one for your target program in the lower 1 megabyte of memory. MagicCV runs in extended memory as a protected mode kernel managing the two virtual environments. Only 4–8K of conventional memory is required as a communications area between the two virtual machines. The slick aspect of MagicCV is that all patching of CodeView is done dynamically using the memory protection features on the 80386 in the protected mode. New versions of CodeView may not require changes to MagicCV. It also should be relatively easy for Nu-Mega to produce an equivalent product for the new symbolic debugger expected from Borland this fall.

System requirements are an 80386 machine with at least 384K of extended memory above 1 megabyte although 640K of extended memory is preferred. Documentation is a short, 37-page manual with an index and includes an overview, installation details, instructions for the use of MagicCV with Soft-ICE, a section on the theory of operation, and error messages. Since MagicCV is a protected mode program, it cannot coexist with other protected mode software programs; however, MagicCV can coexist with other programs that use extended memory (RAMDRIVE and VDISK) and that make only BIOS calls. MagicCV may be used with Nu-Mega's Soft-ICE debugger ($386 if purchased separately, both products for $499). This allows you to run CodeView or some other tool as your primary debugger and to pop into Soft-ICE to set sophisticated breakpoints. This is the debugging environment Microsoft should have provided.

## Translators

**FOR_C**
Cobalt Blue
1683 Milroy, Suite 101
San Jose, CA 95124
(408) 723-0474

Cobalt Blue has issued their first release of FOR_C, their new FORTRAN to C translator ($750). Cobalt Blue previously offered a RATFOR to C translator that also supported parts of the FORTRAN language (see *PJ* 6.2). The new release, though, is a completely different product written from the ground up more as a compiler although there are some similarities with the earlier product. FOR_C is designed for experienced computer programmers who want to translate FORTRAN programs to C. It was developed for "one-




PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# LIBRARY SUPPORT.

# ANOTHER CAREER OPPORTUNITY FOR PCs *WITHOUT* OUR 386 MOTHERBOARD.™

***With* Our MotherBoard, Your PC, XT or AT Can Run New OS/2 Releases!** If you'd rather keep your computer on your desk than under foot, get our 386 MotherBoard. Our Board will give your spreadsheet, complex CAD and graphics, and multitasking applications a boost to boast about. In fact, our line of MotherBoards accelerate your PC, PC/XT, PC/AT or clone to speeds equal to the IBM PS2 Model 80. And faster. Because we've built in 1 Megabyte of high speed RAM and a 387 math coprocessor socket.

**Accelerator Cards Have No Future.** An accelerator card can only turn an obsolete computer into a faster obsolete computer. Our MotherBoards advance your computer into a new 386 system! They run OS/2, Windows/386, UNIX V and PC-MOS/386. For more power, you'll find 16-bit expansion slots that accommodate the latest I/O expansion cards. **Only our 386 MotherBoards give your PCs an unlimited future!**

**The Critics Applaude!** *PC Magazine* awarded our Board "The Editor's Choice" for 386 Replacement Boards. *PC World* called it "the Upgrade Product of the Year."

**Technical Features** ▪ 16 MHz 80386 ▪ 1 Megabyte of 100 nsec 4-way interleaved RAM ▪ PC/AT compatible I/O and BIOS for support of OS/2 ▪ Six 8-bit expansion slots ▪ two 16-bit expansion slots; four on 386 MotherBoard/AT ▪ One 32-bit expansion slot for up to 12 Megabytes of high speed memory ▪ Battery-powered clock calendar.

| | |
|---|---|
| 386 MotherBoard/PC or MotherBoard/XT | $1495 |
| 386 MotherBoard/AT | $1595 |
| 32-bit RAM Board (2 Mbyte installed; up to 4 Mbytes) | $495 |
| 16 MHz 80387 math coprocessor | $695 |
| 16-bit combination hard disk/floppy disk controller | $245 |

Step up your PCs with our easy-to-install 386 MotherBoards. For more information, call: **1 (800) 443-6284**. In New York: **(516) 434-1600**. All our products are available at your local computer dealer!

Hauppauge Computer Works, Inc.
175 Commerce Drive,
Hauppauge, New York 11788

Hauppauge!

Number **122** on the Reader Service Card

Trademarks: IBM PC, XT, AT and PS2: IBM.

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

time" translations of clean FORTRAN code (debugged and running) although it can also be used for maintaining both FORTRAN and C versions of the same code. The source code supported by the translator is standard FORTRAN-77 (space sensitive) although several common extensions are supported (inline comments, tab formatting, VAX DO WHILE and ENDDO, INCLUDE, and MILSPEC intrinsic functions). The translated C code is designed for a C compiler with ANSI C compatibility. The supplied C runtime library code to support the translation makes calls to the standard ANSI C library functions. The runtime library is supplied as source code for compiling with your favorite compiler. A make file is included to create the library using Microsoft C 5.x.

The translations made by FOR_C are generally literal with the exception of character strings, which are converted to null terminated C strings, and a few minor optimizations. The translated C code will look a lot like the original FORTRAN. The translator reads in a macro file of default translation macros (FORC.MAC), followed by custom macros for the file being translated, and a C prototype file for the FORTRAN source being translated. The macro files change FORTRAN variables and routines that use C keywords, allow tailoring the function call mechanism to call by value or address, and provide a mechanism to handle non-standard FORTRAN extensions or special features. The translator can generate a C prototype file from the FORTRAN source deck for subsequent use in the translation process. All source comments are converted to C style comments. FORTRAN I/O is translated, but this is a bit of a kludge, as calls to C functions that emulate FORTRAN I/O (printf and scanf calls) are not generated directly.

FOR_C supports a number of command-line options. These include options for treating text past column 73, for mapping a "D" (VAX debug) character in column 1, for whether the C target *ints* are 2 bytes or 4 bytes (generates C shorts or longs), for promoting FORTRAN integer and logical variables to 4 bytes, for not promoting functions and arguments passed by value (that is, floats don't change to doubles), for treating do loops as if under FORTRAN 66 and declaring all variables static, for whether to pass intrinsic function arguments by address or value, and for generating a C prototype file, optimizing common declarations, deleting unused variables, and taking options from a response file. FOR_C has a built in C-like preprocessor that can be used in the macro files or the original FORTRAN source code. The current version of FOR_C translates each module in its entirety in one pass. There are limits on the number of names, format statements, assigned variables, and common blocks in a module. Also, I found that FOR_C would run out of memory space while translating large modules (2000 line routines).

One holdover from the earlier product is the way translation errors are handled. If FOR_C hits an error, the translation process stops. You must remove the cause of the error or modify one of the macro files to pass the statement through as a literal to continue the process. Fortunately, the translator zipped through the FORTRAN source I fed it with only a few errors. That is not to say that the generated C code doesn't have some errors. As noted earlier, FORTRAN I/O is handled a bit awkwardly. Quite often you might want to rip out the original FORTRAN I/O anyway. The generated C code usually compiles with only warnings. The format of the C output could be improved, but this could be done with a separate C beautifier program. All in all, the translator is usable now (except for the translation of large modules). The program author is actively working on major improvements of the translated I/O code for the next release due this fall. The FOR_C package is provided on two 360K diskettes, one with the translator and its macro files and a second runtime library source code disk. Technical support is available by phone and mail. ♦


## Software Developers

### *We need your program!*

Do you have a program that's good enough to sell, but don't want the problems or financial risk of producing, typesetting, printing, packaging, warehousing, marketing, distributing and supporting a product?

Why start your own software house?
We've done it for you!

**Merlin Publishing Group**
is now accepting submissions of micro-computer software for publication.

You get:
- initial cash payments
- generous royalties
- to spend your time programming

But we can't help you if you don't submit. Call or write today for our submission guideline kit.

MERLIN
PUBLISHING GROUP ™
1240 Johnsons Ferry Place, Suite A10
Marietta, GA 30068
(404) 977-6034

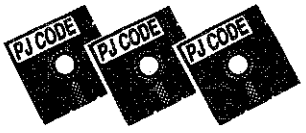
Number 38 on the Reader Service Card



## *PJ Code Is Available!*

On disk from
**Programmer's Journal**
*Call toll free, 800/234-0386*
*See ad on page 89*

On BBS from
**FIDO/OPUS Network**
*Blazer Net 152 Node 9*
*(503) 343-9724*
and
**Turbo Source Search**
*P.O. Box 876*
*Scituate, MA 02066*
*Voice (617) 545-6677*
*Data (617) 545-9131*




All Programmer's Journal subscribers are pre-registered in the Turbo Source Search callers' log. To download PJ code,
Have your modem call (617) 545-9131. At the prompt, type in your first and last names (no middle initial, no punctuation). At the next prompt, type in your password (the first four numbers of the subscription ID number on your PJ mailing label).
At the command prompt, type J (for Join); then type 3 for the PJ Conference.
Once in the PJ Conference, type F (for File Listings) to view the sub-directory menu. Type in the number of the sub-directory you want to see and note the name(s) of the file(s) you want to download.
Type D (for Download). At the prompt, type in the name of the file(s) you want to download.
For help, type H and then the key letter from the main menu.
For further help, leave a message for SYSOP.
To unarchive the files you download, use file PK36.EXE (Phil Katz's excellent archive tools). We recommend that you use ARCM240.ARC (John Newlin's ARCMASTER) in conjunction with PK36.EXE.
Besides PJ's code, Turbo Source Search contains over 100MB of files related to programming.





PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

## LOW COST ADVERTISING

If you sell products to or provide services for professional programmers, a PJ Billboard ad is an easy, low cost way to reach more potential buyers. Regular listings include a headline, company name, address and phone number, Reader Service number and approximately 400 characters of sales pitch. Display listings are 1 5/8" wide by 2 3/8" deep. Costs are $125 for a single insertion, $115 per insertion for 3 insertions and $95 per insertion for 6 consecutive insertions. PJ Billboard ads must be prepaid and are non-commissionable. Call the ad department for deadlines.

**Programmer's Journal**
*P.O. Box 30160*
*Eugene, OR 97403*
*503-747-0800*

## SEE CURRENT FILE ACTIVITY

WATCH I-O is a pop-up utility that displays files open, DX:DS of open and last read/write, last read/write counts, last Exec, last file opened, last file-not-found. User can select hot key, pop-up for file opens, pop-up for file-not found, track file opens on printer.

WATCH I-O is only $49.95. With anti-virus option $69.95. VISA/MC.

**Quantum Publishing**
*23860 Miles Road*
*Cleveland, OH 44128*
*800-342-6237* **RS# 201**

## UNDERSTANDING C VIDEO TAPE

Get guaranteed results with this three volume video course! For expert or novice—these tapes (total 6 hours) cover ANSI C for today's professional programmer. Loaded with examples, these tapes cover it all! Handy reference book included.

$59.95 postpaid. Add appropriate tax in TX. Send check or money order to:

**AUTOMATION EDUCATION**
*2309 Royce Drive*
*Arlington, TX 76016-1232*
*817-496-4125* **RS# 207**

## PL/M TO C TRANSLATOR

LISP-based translator for major assistance in translating from INTEL PL/M to C language. Human readability a high priority. Sample listings available. Runs under MS-DOS 2 and up. Ports possible for VAX with VMS or BSD 4.2.

$5000 site license includes 3 months of telephone support.

**Frank Korzeniewski Consulting**
*1564-A Fitzgerald Drive #137*
*Pinole, CA 94564*
*415-799-1819* **RS# 209**

## AT LAST! A C SOURCE CODE ANALYSIS TOOL WITH FULL PREPROCESSING CAPABILITY

C:LINES/C:TREE features

- ANSI draft compatible preprocessor
- Outlined listings showing flow of control within procedures
- Tree diagrams showing flow of control between procedures
- Source code reformatter

$79.95, shipping $3 domestic, $15 overseas. MD residents add 5% sales tax. Requires PC compatible, 256K.

**SoftRex**
*4807 Bethesda Ave. #287*
*Bethesda, MD 20814*
*301-881-8274* **RS# 185**

## INTELLIGENT REVERSE-ASSEMBLER

MASTER*KEY is an intelligent self-documenting MS-DOS disassembler that returns fully documented formatted assembly language source code (MASM 5.0 compatible) from ANY binary file of ANY size. It works from the command line or interactively with menus. Selective partial output speeds through a program file keeping listings and source files down to a manageable size. MASTER*KEY's cross-reference locates all items by memory location in the program file.

$79.95 + 2.50 shipping. Add 6 1/2% sales tax in CA.

**Sharpe Systems Corporation**
*2320 E. Street, Dept. 1610*
*La Verne, CA 91750*
*714-624-0346* **RS# 175**

## PROGRAM FASTER, INSTANTLY

EXPERT KEY automatically types variables and words at the touch of a single hot key. No macro keys to program or remember.

Memory resident program learns YOUR words from the keyboard and screen. Start typing and a highlighted word pops up. Hit the hot key and the word is completed or keep typing and the word is gone. Variable activity level. Program learns editing sequences and can recall previous command lines. Tracks keystrokes typed and saved. Runs with your editor, DOS and other applications.

LIMITED OFFER $65 (Reg. $95). Add $3 S&H. CA residents add 6%. For PC/XT/AT with 30 day guarantee.

**Newell Technologies, Inc.**
*27758 Santa Margarita Pwky. #330C*
*Mission Viejo, CA 92691*
*714-583-7848* **RS# 203**

**PJ ISSUE SPECIAL!**
See ad on page 89 for details

## OPTASM, THE FASTEST MACRO ASSEMBLER AVAILABLE

Processing 75M lines/minute, OPTASM is a macro assembler (Source Code compatible with MASM) that is loaded with features.

- Works with Borland and Microsoft languages
- Demo: FREE from SLR Systems's BBS (412-282-2799) or send $10 to SLR Systems and we'll send you a disk.
- Now available with CodeView support

**OPTASM $125 (list) CALL TO ORDER or FOR MORE INFO: 800-833-3061**

**SLR Systems**
*1622 North Main*
*Butler, PA 16001*
*412-282-0864* **RS# 165**

## NATURAL LANGUAGE FOR YOUR DBASE APPLICATIONS

What would YOUR customers pay for a natural language query interface for their dBASE/FoxBase applications? With ELLIE you can build memory-resident ad hoc query capabilities into any (interpreted mode/non-compiled) Dbase language system. Rated "Excellent" by USER (University Software Evaluation Report) for both esae-of-setup and user-friendliness (3.6 out of a 4 overall rating). Full-featured, with transformational grammar processing, automated installation/dictionary-building, support for synonyms, antonyms, phrase definition, unit conversions, macro handling and more. Your cost: $119 plus $50 per site/client runtime license fee. Your profit opportunity: UNLIMITTED. And there's a full 30-day money back guarantee.

**ELF Software Company**
*210 West 101 Street*
*New York, NY 10025*
*212-316-9078* **RS# 113**

## FUTURE86, THE LANGUAGE

Variable level, extensible, rommable language gives complete control. For MS/DOS or targeted use. Use for any 8086 class cpu including 80186/88, V20, V25/35, 80386 etc.

Simple syntax. Easy programming. Compact code. Especially ideal for real time control applications. Easily optimized for speed or size. Interactive debugging facilities. User Shell. Complete library support. Library source available. Coming soon... REM86 remote target debugger.

Investigate FUTURE86... World Class! Used in thousands of varied and demanding applications. Prices start at $349.

**Development Associates**
*1520 South Lyon*
*Santa Ana, CA 92705*
*714-835-9512* **RS#225**

**PERSONAL MEASURE**

measures PC performance. Use to tune system, evaluate products, do benchmarks. TSR less than *10K* & self removing. Reports w/ menus & graphics. DOS 2.0+; CGA, EGA, VGA, Hercules graphics; FX-80, Proprinter, LaserJet. *$69.95*, + $2 S&H, $5 for 3 1/2" media, 5% tax for MA residents. Sensible 30-day money back guarantee.

VISA MC

**Spirit of Performance, Inc.**
*73 Westcott Road*
*Harvard, MA 01451*
*508-456-3889*
*MCI Mail 357-1400*

Number **197** on the Reader Service Card

## $50 INTRODUCTORY OFFER!

VM/386 enables you to turn one 386 PC into several by creating "virtual machines" that run REAL PC-DOS & your current software. You qualify if you are a Fortune 1000 MIS Director, VAR, Systems Integrator, Computer Dealer or Software Developer. To take advantage of this offer, send your business card & check for $55 (includes shipping) to IGC. CA residents add applicable sales tax. Or call for information, but HURRY - OFFER EXPIRES 10/15/88.

**IGC**
*4800 Great America Parkway*
*Santa Clara, CA 95054-1221*
*408-986-8373*
*Outside CA*
*800-458-9108* **RS# 169**

Computer & Technical
*** BOOKS ***
from
INFOSOURCE
International
*PROGRAMMER'S PC SOURCEBOOK*
$24.95 +S/H
Box 238, Dept. PJ 988
Barryville, NY 12719-0238
*1-800-327-7970*

Number **217** on the Reader Service Card



PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# ADVERTISER INDEX

| Reader Service | Advertiser Name | Page Number |
|---|---|---|
| 220 | Aesoft Corporation | 30 |
| 180 | A.I. Architects, Inc. | 33 |
| 218 | Alsys, Inc. | 71 |
| 215 | American Cosmotron | 11 |
| 184 | American Cybernetics | 38 |
| 162 | Anji Software, Inc. | 20 |
| 170 | ASCII | 21 |
| 86 | Aspen Systems | 26 |
| 30 | Austin Code Works | 49 |
| 207 | Automation Education | 95 |
| 51 | Az-Tech Software, Inc. | 74 |
| 3 | Borland International | C4 |
| 219 | Byte Design LTD. | 21 |
| 38 | C Ware Corporation | 74 |
| 56 | CompuView Products, Inc. | 9 |
| 224 | Copy Technologies | 59 |
| * | Creative Programming Consultants | 46 |
| 5 | Dair Computer Systems | 14 |
| 225 | Development Associates | 95 |
| 8 | Ecosoft, Inc. | 53 |
| 173 | Elan Computer Group, Inc. | 91 |
| 113 | ELF Software Co. | 95 |
| 206 | Flambeaux Software | 82 |
| 118 | Fleming Software | 82 |
| 209 | Frank Korzeniewski Consulting | 95 |
| 135 | Gilmore Systems | 76 |
| * | Gimpel Software | 56 |
| 204 | Guidelines Software, Inc. | 15 |
| 122 | Hauppauge Computer Works, Inc. | 93 |
| 154 | HavenTree Software LTD | 19 |
| 169 | IGC | 95 |
| 217 | Infosource International | 95 |
| 202 | Javert, Inc. | 14 |
| 211 | Kadak Products LTD. | 75 |
| 155 | Komputerwerk | 25 |
| 63 | Laboratory Microsystems, Inc. | 83 |
| 212 | Magna Carta Software | 48 |
| 223 | Marietta Systems | 76 |
| 213 | Max Software Consultants | 43 |
| 141 | Merlin Publishing Group | 94 |
| 143 | Metagraphics Software Corporation | 90 |
| 214 | Micro Interfaces Corporation | 92 |
| 45 | MetaWare, Inc. | 1 |
| * | PLEASE CONTACT THESE ADVERTISERS DIRECTLY | |

| Reader Service# | Advertiser Name | Page Number |
|---|---|---|
| 121 | MISOSYS Inc. | 88 |
| 94 | MMC AD Systems | 52 |
| 109 | NanoSoft | 15 |
| 203 | Newell Technologies, Inc. | 95 |
| 160 | Nu-mega Technologies | 3 |
| 16 | Opt-Tech Data Processing | 73 |
| 195 | Paradigm Systems | 62 |
| 145 | Pathfinder Associates | 54 |
| 208 | PC Tech, Inc. | 27 |
| 216 | Peacock Systems | 81 |
| 6 | Periscope Company, The | 84 |
| 52 | Powerline, Inc. | 47 |
| 26 | Polytron Corporation | 5 |
| * | *Programmer's Journal* Mail Order | 89 |
| 200 | Programmer's Paradise | 7 |
| 199 | Prospero Software, Inc. | 36-37 |
| 189 | QCAD Systems | 80 |
| 97 | Qualstar Corporation | 80 |
| 201 | Quantum Publishing | 95 |
| 147 | Quarterdeck Office Systems | C2 |
| 222 | Rainbow Technologies | 77 |
| 20 | Rational Systems, Inc. | 55 |
| 198 | Research Group, The | 41 |
| 148 | SemWare | 50 |
| 175 | Sharpe Systems Corporation | 95 |
| 186 | Silver State Software | 63 |
| 165 | SLR Systems | 95 |
| 177 | Soft Evolutions, Inc. | 26 |
| 185 | SoftRex | 95 |
| 167 | Software Development '89 | 31 |
| 98 | Software Family, The | 51 |
| 221 | Software Security | 67 |
| 197 | Spirit of Performance, Inc. | 95 |
| 158 | Summit Information Systems, Inc. | 18 |
| 85 | Sunny Hill Software | 91 |
| 171 | Systemat Corporation | 8 |
| 166 | Tool Makers, The | 88 |
| 25 | TurboPower Software | 39 |
| 35 | Turbo Source Search | 43 |
| 125 | V-Communications | 28 |
| 196 | West Chester Group | 25 |
| 123 | Xenosoft | 75 |
| 182 | Zortech, Inc. | C3 |
| * | PLEASE CONTACT THESE ADVERTISERS DIRECTLY | |

When contacting any of the above advertisers, please tell them you saw their product advertised in ***Programmer's Journal*** and that you appreciate their continuing support of this magazine.


# PJ ISSUE SPECIAL!

**The Ray Duncan Series—**
**Starting Your Own Software House**
**PJ 5.1 Jan/Feb 87**
**PJ 5.2 Mar/Apr 87**
**PJ 5.3 May/Jun 87**
Regularly **$21.00**, these 3 popular back issues can now be yours for only **$14.95!**
See house ad, page 89 for details

# ADVERTISE IN
***Programmer's Journal***
**Nov/Dec Issue, Volume 6.6**

| Space Reservation Deadline: | Materials Deadline: |
|---|---|
| **September 9, 1988** | **September 23, 1988** |

Call ***Programmer's Journal*** Ad Department to reserve your space today!
**(503) 747-0800**



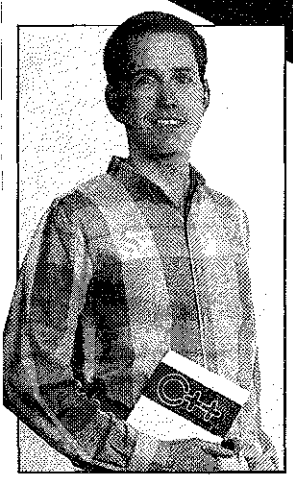


PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

INTRODUCTORY PRICE

**$99.95**

Regular price $149.95.



WALTER BRIGHT – Programmer

*"This is a revolutionary product which reflects the fact that Walter Bright is one of the world's best programmers."*
Andy Redfern, Personal Computer World

*"I find it amazing that Walter continues to find ways to improve his already excellent product."*
Scott Robert Ladd, Micro Cornucopia

FROM ZORTECH INC.

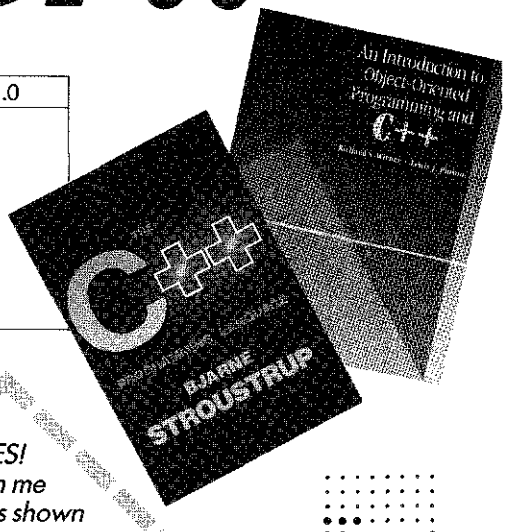
# C is great!
# C++ is better!!

Performance benchmarks

| Benchmark | Zortech C | Zortech C++ | Turbo C 1.5 | Quick C 1.0 |
|---|---|---|---|---|
| Sieve | 20.49 | 20.54 | 23.62 | 22.72 |
| Rsieve | 20.49 | 20.54 | 23.62 | 22.03 |
| Integer | 1.32 | 1.38 | 6.31 | 6.49 |
| Float | 0.17 | 0.22 | 52.29 | 51.03 |
| *Float | 32.73 | 37.74 | 52.39 | 51.63 |
| Pointer | 17.91 | 17.96 | 17.13 | 16.87 |
| Rpointer | 17.79 | 17.91 | 17.14 | 16.64 |
| Loop | 3.90 | 3.90 | 3.90 | 3.90 |
| Optimize | 0.49 | 0.60 | 8.46 | 8.79 |

Benchmarks were run on an 80286 based IBM compatible at 6Mhz with no 8087.

*The float benchmark was re-run without optimization.




***Zortech C++ is the first native-code C++ compiler for MS-DOS compatible machines. This means you don't need to spend a lot of money on a C compiler. In fact, Zortech C++ comes with a C++ compiler, a C compiler, a linker, a librarian, a fully integrated editing environment, context sensitive help and the fastest graphics library you've ever seen!***

***With Zortech C++, you can mix and match code in C++, C and assembler and link them all together with a single command!***

***Zortech C++ comes with a complete C++ and C library, including compatibility with the ANSI C draft. Unique to Zortech C++ is support for mice, fast screen writes, re-entrant floating point and speaker support.***

***Now you can move your Microsoft and Turbo C code over to Zortech C++ – we've included compatible library functions! Also, we have made Zortech C++ compatible with Microsoft's Codeview debugger!***

***Enter the world of Object Oriented Programming (oops) with Zortech C++!***

***Sure, C is great – but C++ is better!***

Number 182 on the Reader Service Card

*YES!*
*Rush me C++ as shown below:*

☐ *Zortech C++ $99.95**

☐ *C++ Book $29.95** ☐ *Intro C++ Book $27.95**

*VISA/MC/COD CHECK ACCEPTED*

ZORTECH

*Name*.....

*Address*.....

..... *Phone*.....

*VISA or MC#*..... *Exp. date*.....

*To: ZORTECH INC. 366 Massachusetts Ave., Arlington, MA 02174. Tel: 617-646-6703. Fax: 617-643-7969.*

**CALL THE ORDER HOTLINE 1-800-848-8408**

**Prices do not include shipping*

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED

# "Behind the beauty of the Turbo C environment stands the brawn of a full-fledged compiler"

*Marty Franz, PC Tech Journal*

"Taking compilers and program development tools into the next generation is Borland International's Turbo C, a $99.95 package that will stun you with in-RAM compilations that operate at warp speed.

. . . a 21st century compiler at a preinflation 1967 price. Is it any wonder that Turbo C was included in the Best of 1987?

*Richard Hale Shaw, PC Magazine*

Turbo C represents an all-new price-performance level—one that will be hard to match, much less beat.

*Stephen Randy Davis, PC Magazine*

Turbo C showed excellent compiler speeds, good overall benchmark scores, and extraordinary floating-point performance.

*Scott Robert Ladd, Micro Cornucopia*"

Actual photograph of Turbo C graphics displayed on IBM 8514 screen.*

## *Our new Turbo C 1.5 is a technological tour de force*

At Borland we believe the slow way is no way, so Turbo C® is a racer. And as well as white-knuckle speed, Turbo C also gives you spectacular graphics.

Number 3 on theReader Service Card

## *Some of the reasons why the critics are so enthusiastic about Turbo C 1.5*

Turbo C now includes:

- A professional-quality graphics library of over 70 functions
- A librarian that allows you to build your own object module libraries
- Context-sensitive help for the language and the library routines
- Text/video functions, including windows
- 43- and 50-line mode support
- VGA, CGA, EGA, Hercules, and IBM 8514 support
- File search utility (GREP)
- Sample graphics applications
- More than 100 new functions

## *The professional optimizing compiler for less than $100.00*

For professional-quality C at a sane price, nothing comes close to Turbo C. It's super-fast and super-graphic. (We used it ourselves to write Eureka:™ The Solver and to develop the presentation-quality graphics in Quattro,™ our new and highly successful professional spreadsheet.) No one can deliver technical superiority like Borland.

*60-Day Money-back Guarantee***

For the dealer nearest you,
**Call (800) 543-7543**

BORLAND INTERNATIONAL

***Minimum system requirements:*** For the IBM PS/2™ and the IBM® family of personal computers and all 100% compatibles. PC-DOS (MS-DOS®) 2.0 or later. 384K.
*Artwork metafile courtesy of Genigraphics® Corporation
**Customer satisfaction is our main concern; if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.

All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Copyright ©1987 Borland International, Inc. BI 1221

PRODUCED 2004 BY UNZ.ORG
ELECTRONIC REPRODUCTION PROHIBITED